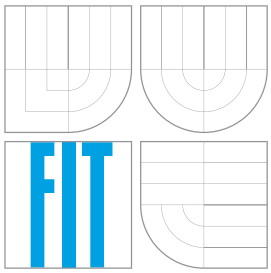
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

GEOGRAFICKÝ INFORMAČNÍ SYSTÉM PRO PASPORTIZACI A VIZUALIZACI ROZLEHLÝCH POČÍTAČOVÝCH SÍTÍ

GEO INFORMATION SYSTEM FOR PASSPORTIZATION AND VISUALIZATION OF WIDE COMPUTER NETWORKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. ONDŘEJ FIBICH
AUTHOR

VEDOUCÍ PRÁCE — doc. Dr. Ing. DUŠAN KOLÁŘ
SUPERVISOR

BRNO 2014

Abstrakt

Práce je zaměřena na problematiku pasportizace a vizualizace rozlehlých počítačových sítí pomocí geografických informačních systémů. Seznamuje čtenáře s principy a vlastnostmi geografických informačních systémů a prostorových databází, které často slouží jako nosiče dat geografických informačních systémů. Hlavní část práce je věnována pasportizaci počítačových sítí se zaměřením na fyzickou strukturu sítí. Po stanovení formátu pasportu se práce věnuje analýze, návrhu, implementaci a testování geografického informačního systému určeného pro evidenci a vizualizaci pasportu v tomto formátu. Práce je zakončena případovou studií demonstrující postup pasportizace přístupové počítačové sítě vytvořeným systémem.

Abstract

The thesis is focused on the problem of the passportization and the visualization of wide computer networks with the usage of geographic information systems. It discusses principals and properties of geographic information systems and spatial databases that are frequently used as data carriers for geographic information systems. The main part of the thesis is dedicated to the passportization of computer networks with focusing on the physical network structure. After the definition of the passport format the thesis continues with the analysis, the proposal, the implementation and the testing of the geographic information system for managing and visualization of the passport in this format. The thesis is ended by a case study that demonstrates the passportization process of an access computer network in the implemented system.

Klíčová slova

Pasportizace, počítačové sítě, GIS, prostorové databáze, mapový server, OpenLayers, PostGIS, GeoServer, Java EE, JAX-RS, EJB, JPA, jQuery.

Keywords

Passportization, computer networks, GIS, spatial database, map server, OpenLayers, PostGIS, GeoServer, Java EE, JAX-RS, EJB, JPA, jQuery.

Citace

Ondřej Fibich: Geografický informační systém pro pasportizaci a vizualizaci rozlehlých počítačových sítí, diplomová práce, Brno, FIT VUT v Brně, 2014

Geografický informační systém pro pasportizaci a vizualizaci rozlehlých počítačových sítí

Prohlášení

Čestně prohlašuji, že jsem tuto semestrální práci vypracoval samostatně pod vedením pana doc. Dr. Ing. Dušana Koláře. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Ondřej Fibich
26. května 2014

Poděkování

Rád bych poděkoval panu doc. Dr. Ing. Dušanu Koláři za trpělivé vedení práce a odbornému konzultantovi panu Ing. Tomáši Dulíkovi, Ph.D. za poskytnuté vědomosti a pomoc při tvorbě práce.

Obsah

Kapitola 1

Úvod

V současné době (duben 2014) většina organizací, které budují optické, metalické a rádiové sítě většího rozsahu, pociťuje nutnost dokumentovat jejich přesný popis fyzické a logické struktury. Důvodů může být několik, avšak tím hlavním je přehledná správa celé infrastruktury.

Fyzický popis sítě je u menších subjektů obvykle dokumentován jednoduchými vektorovými editory nebo CAD systémy, výsledná data jsou ovšem obtížně udržovatelná a nelze je použít v kombinaci s jinými systémy. Větší subjekty za tímto účelem často investují nemalé finanční prostředky do proprietárních systémů.

Pro logický popis sítě se u mnohých menších firem a jiných podobných subjektů bohužel užívají programy pro tvorbu tabulek z různých kancelářských balíků (např. Microsoft Excel). Situace je zde ve srovnání s fyzickým popisem sítě většinou mnohem lepší, jelikož na částečný logický popis sítě bývá obecně napojen systém účtování za poskytované síťové služby, monitorování atd. Pro tyto účely existuje několik proprietárních a svobodných informačních systémů s různými vlastnostmi a funkcionalitou.

Celkově vzato neexistuje řešení, které by bez jakéhokoli omezení definovaného v tomto dokumentu dokázalo oba směry popisu sjednotit do jednoho uceleného systému. Tento fakt nutí subjekty budující počítačové sítě dokumentovat svoji síťovou infrastrukturu na dvou různých místech, což je značně neefektivní. Z těchto důvodů byl navrhnut vývoj systému, který bude uvedené aspekty splňovat.

Cílem této práce je analyzovat, navrhnout a implementovat systém pro dokumentaci fyzického popisu počítačových sítí, který bude v budoucnosti možné napojit na některý z existujících systémů pro dokumentaci logického popisu sítě. Vytvoření systému bylo zadáno firmou *INTERNEXT 2000, s.r.o.* a podpořeno *Inovačním voucherem ve Zlínském kraji – etapa II.*

Systém pro dokumentaci fyzického popisu pracuje s geografickými polohami jednotlivých prvků sítě, a proto je kategorizován do skupiny geografických informačních systémů. Geografickým informačním systémům se podrobně věnuje kapitola 2. Kapitola 3 obsahuje stručný úvod do prostorových databázových systémů, které často slouží jako nosiče dat pro geografické informační systémy. Dokumentace fyzické struktury počítačových sítí je do jisté míry společná pro dokumentaci inženýrských sítí ve stavebnictví. Odborně tuto dokumentaci nazýváme pasportem a proces tvorby dokumentace pasportizací. Geografický informační systém pro dokumentaci fyzické struktury počítačové sítě lze tedy nazvat geografickým systémem pro pasportizaci počítačových sítí. Ve stavebnictví je pevně stanovený formát pasportu zákonem, v oboru počítačových sítí takový formát neexistuje, a proto kapitola 4 shromažďuje informace spojené s problematikou počítačových sítí a následně vytváří

vlastní formát. Po definici formátu pasportu lze přistoupit k analýze, návrhu a testování systému pro správu a vizualizaci pasportu. Postupně jsou těmto problematikám věnovány kapitoly 5, 6 a 7. Pro demonstraci funkčnosti vytvořeného systému je v kapitole 8 zhotovena případová studie. V závěrečné kapitole 9 je celá práce spolu s jejími výsledky shrnuta a jsou navrženy vhodné body pro její rozšíření a vylepšení.

Diplomová práce navazuje na semestrální projekt vypracovaný v totožném akademickém roce. Diplomová práce je oproti semetrálnímu projektu rozšířena o kapitoly 6, 7 a 8. Kapitola 5 byla v diplomové práci z velké části modifikována tak, aby reflektovala změny požadavků zadavatele, které vznikly po odevzdání semestrálního projektu.

Kapitola 2

Geografické informační systémy

Geografický informační systém (GIS), jak již z jeho názvu vyplývá, je specifický *informační systém*. Pokud bereme v potaz definici informačního systému jako souboru komponent, které shromažďují, zpracovávají, vyhodnocují, uchovávají a distribuují informace pro podporu rozhodování, prezentaci a koordinaci v jisté organizaci [33, s. 6], pak specifičnost GISů oproti běžným informačním systémům spočívá v jejich vyčlenění pro zpracování geografických informací. I když existuje větší množství různých typů informačních systémů, GIS je v určitých vlastnostech a principech natolik odlišný od jiných informačních systémů, že vzniklo samostatné vědní odvětví zabývající se pouze GISy. Tento fakt dokazuje unikátnost a důležitost GISů.

GIS bývají často mylně zaměňovány se systémy pro vizualizaci map, jelikož mapové výstupy bývají nejčastějším prvkem interakce mezi uživatelem GISu a jeho rozhraním. Z omylu vyvádí definice GIS dle Vojtička [61], která udává, že vizualizace je pouze jednou z funkcí GIS systémů:

> *GIS je organizovaný soubor počítačového hardware, software, geografických údajů a personálu určený k efektivnímu sběru, uchovávání, obnovování, manipulaci, analýze a zobrazování všech geograficky vztažených informací.*

Předmětem GIS systémů není žádná činnost neelektronického charakteru. Příkladem takové činnosti může být geodetické měření prováděné v terénu. *Geografická informace (geoinformace)*, kterou dle výše uvedené definice GIS zpracovává, je informace o objektu nebo jevu, jejíž součástí musí být mimo jiné její umístění vztažené k Zemi. Zmiňovaná součást je označována jako *geografická poloha*. Práce s geografickou polohou není triviální záležitostí, poněvadž je nutné zavést *geografické souřadnice* pro jednoznačné určení polohy na povrchu Země [24]. Jelikož je GIS elektronickým systémem, musí existovat možnost pracovat s koordináty elektronicky. Tento problém je popsán v kapitole 2.1.

GIS se jako mladá vědní disciplína rychle rozvíjí především kvůli širokému uplatnění v různých odvětvích lidských činností (např. armáda, státní správa, zemědělství, geologie, meteorologie). Pro použití či tvorbu GIS je nutné pochopit způsob reprezentace a uložení dat (viz kapitola 2.1), jak data vstupují a vystupují do/ze systému (viz kapitola 2.2) a jak lze data obsažená v systému analyzovat (viz kapitola 2.3).

2.1 Geografická data

Geografická data (geodata) jsou počítačově zpracovatelnou formou geoinformací, která obsahují implicitní nebo explicitní určení geografické polohy ve vztahu k Zemi (místu na Zemi). Úkolem geodat je identifikace geografické polohy, charakteristika jevů a hranic mezi

nimi [50]. Geodata jsou souborem *geografických objektů (geoobjektů)*, které jsou složeny z [10, 24]:

- identifikace – rozlišující objekt jednoznačně od jiných,
- prostorové informace – udávající pozici, velikost, tvar a vztahy s ostatními objekty,
- atributů – vkládající popisné informace neprostorového charakteru,
- časových informací – umožňující zaznamenat dynamické vlastnosti dat (nepovinná součást).

Příkladem geoobjektu může být reprezentace řeky, kde je identifikací jedinečné číslo řeky, prostorovou informací zaznamenaná poloha jejího řečiště včetně tvaru říčního koryta a místa stoků s jinými vodními toky, s atributy udávajícími jméno řeky a název dohlížející vodohospodářské organizace, a časovou informací ve formě seznamu zaznamenaných průtoků vody za poslední měsíc.

Vezmeme-li v potaz běžnou mapu státu, objevíme na ní různé informace, které jsou pro zpřehlednění graficky odlišeny. Na první pohled lze například rozlišit řeku od cesty apod. Tento princip je vlastní i GIS systémům. Geoobjekty se snažíme rozdělovat do monotématických souborů. Monotématickým souborem mohou být například všechny vodní toky na zkoumaném území. Soubory nazýváme *mapové vrstvy* (podrobně je jim věnována kapitola 2.1.2).

2.1.1 Geografické souřadnice

Geografické souřadnice (koordináty) slouží pro jednoznačné určení pozice geoobjektu na povrchu Země. Dle způsobu reprezentace pozice pomocí koordinátů rozlišujeme typy pozic geoobjektů na [10, s. 15]:

- Bod – udává pozici jedním koordinátem. Takovým způsobem jsou modelovány pozice geoobjektů, u nichž není nutné měřit žádný rozměr.
- Křivka – je jednodimenzionální geometrický objekt, u kterého lze měřit jeho délku. V GIS systémech využíváme z důvodů náročnosti výpočtů a reprezentace obecné křivky pouze jednu z jejich variant – *lomenou čáru*, která je určena uspořádaným seznamem bodů (koordinátů).
- Plocha – je dvojdimenzionální geometrický objekt. Podobně jako u křivek nepracujeme v GIS systémech se všemi variantami povrchů, ale pouze s *polygony*. Polygon je plocha ohraničená uzavřenou lomenou čárou. Dle [44] může navíc obsahovat polygon také „díry“, které jsou taktéž ohraničeny lomenými čárami. I když polygon s dírami nesplňuje původní geometrickou definici, v GIS systémech se často setkáváme se zaměňováním těchto pojmů.

Je nutné si uvědomit, že způsob reprezentace pozice stejných reálných geoobjektů se může lišit podle požadavků na výsledná data. Běžně se lze například setkat s městy vyznačenými na mapě pouze pomocí bodů nebo naopak přesnými polygony, které města pokrývají.

Pro jednoznačnost koordinátů a definici prostorových vztahů mezi geoobjekty je nutné zavést *souřadný systém*, který pro potřeby GIS systémů umožňuje realizovat vztah mezi

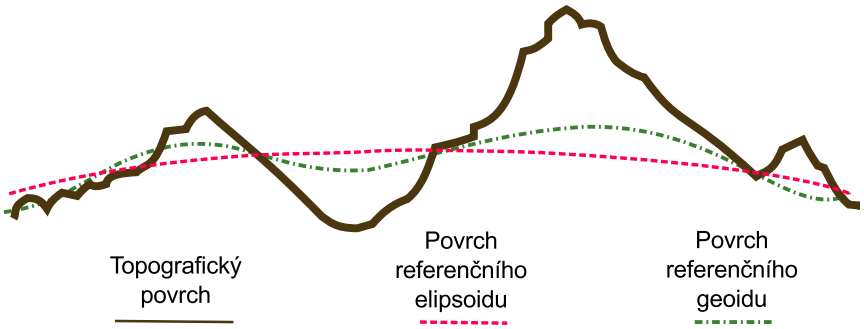
pozicí na povrchu Země a koordinátem geoobjektu [24, s. 19]. Koordinát je poté v daném souřadném systému určen uspořádanou skupinou čísel (vektorem), jejíž mohutnost udává rozměr koordinátu a souřadného systému. Jelikož povrch Země je velmi členitý, je nutné pro vztažení pozice k jejímu povrchu zavést *model povrchu země*, pomocí kterého lze vztažení řešit matematicky.

Model povrchu Země

Pro naprosto přesné určení polohy na povrchu Země je nutné použít trojrozměrný koordinát. Moderní GISy sice tento přístup umožňují, ale práce v systému není intuitivní pro uživatele a zvyšuje složitost topologických vztahů mezi geoobjekty [37]. Běžně se proto setkáváme s dvojrozměrným koordinátem vztaženým ke zjednodušenému modelu povrchu Země.

Nejtriviálnějším modelem Země je koule, ta ovšem nereflektuje zploštění Země v úrovni pólů. V důsledku tohoto faktu byl zaveden přesnější fyzikální způsob popisu povrchu Země – *geoid*. Jedná se o plochu, která do jisté míry kopíruje střední hladinu světových oceánů a moří s prodloužením pod kontinenty [24, s. 15]. Nevýhoda geoidu spočívá v jeho složitém matematickém popisu, a proto byl nahrazen *rotačním elipsoidem*, který za cenu výrazného zjednodušení oproti geoidu umožňuje snadný matematický popis. Namísto termínu rotační elipsoid se v GISech setkáme s pojmenováním *referenční (náhradní) elipsoid*, které lépe vystihuje jeho využití [24, s. 15-17]. Na obrázku 2.1 lze pozorovat podobnost povrchu geoidu a elipsoidu vzhledem k povrchu Země. Dle rozsahu platnosti rozlišujeme náhradní elipsoidy na [24, s. 16]:

- Národní – jsou vztaženy pouze na část zemského povrchu, ve které zaručuje velkou přesnost. Pro Českou republiku je například používán *Besselův elipsoid*.
- Globální – jsou určeny pro modelování celého zemského povrchu, což ovšem způsobuje určení polohy s poměrně velkou chybou. V současné době je nejpoužívanějším globálním náhradním elipsoidem elipsoid navržený pro *WGS-84*.



Obrázek 2.1: Geoid a elipsoid ve vztahu k zemskému povrchu

Souřadné systémy

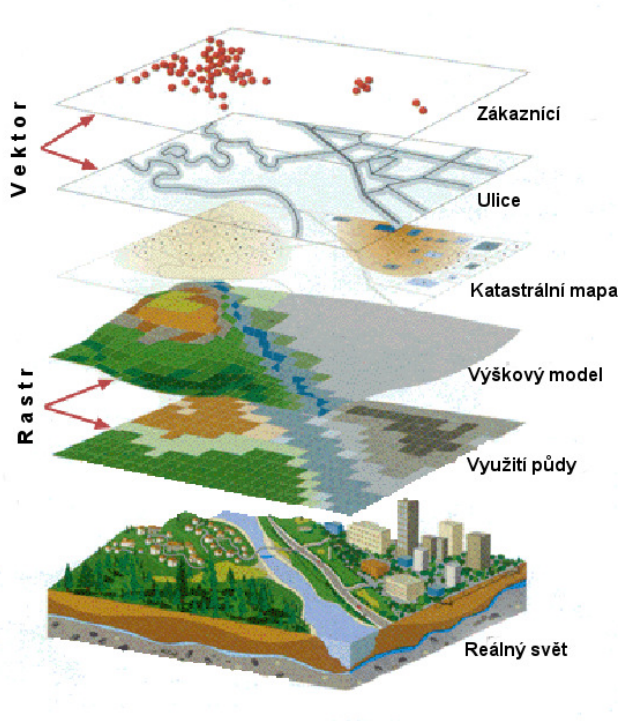
Souřadný (souřadnicový) systém pomocí základních údajů (náhradního elipsoidu) umožňuje jednoznačně definovat měřitelnou polohu. Mezi dvěma polohami definuje *metriku* pro měření vzdálenosti mezi nimi. Příkladem metriky je *euklidovská metrika*, která se využívá v kartézském souřadném systému. *Geografické souřadné systémy* se rozdělují dle použitého náhradního elipsoidu na lokální (národní) a globální. Existuje velké množství dalších souřadných

systémů lokálního i globálního typu, úkolem GIS systémů je umět s těmito souřadnými systémy pracovat a také transformovat geoobjekty z jednoho souřadného systému do jiného [24, s. 17-22]. Mezi souřadné systémy například patří:[1]

- **WGS-84** – je globální souřadný systém typu šířka-délka [24, s. 19], schopný určovat polohu s přesností na metry [6]. Použit je například v systému GPS.
- **S-JTSK** – je lokální souřadný systém používající Besselův elipsoid. Byl vyvinut pro potřeby Československa a v současnosti je používán například v katastrálních mapách vydávaných organizací ČÚZK [24, s. 21]. Vzdálenosti lze počítat pomocí euklidovské metriky a polohu určovat s přesností na centimetry [6].

2.1.2 Mapové vrstvy a jejich reprezentace

Mapové vrstvy sjednocují geoobjekty se stejným tématem do jednoho souboru. Tyto soubory (vrstvy) lze poté v GIS nástrojích libovolně kombinovat a tudíž zjednodušeně modelovat reálný svět. Příklad modelování části reálného světa pomocí mapových vrstev je ukázán na obrázku 2.2.



Obrázek 2.2: Příklad modelování reálného světa pomocí mapových vrstev [24]

Důvodem rozdělení geoobjektů do tématických celků je především snadnější správa, získávání dat, sdílení dat, univerzálnost a jednodušší přístup k analýze dat [24, s. 26]. Při návrhu struktury mapových vrstev je nutné rozhodnout, které geoobjekty je vhodné umístit do samostatné mapové vrstvy a které odlišit pouze atributy.

Pro reprezentaci mapových vrstev se používají dva přístupy známé z oboru počítačové grafiky. Jedná se o reprezentaci mapových vrstev pomocí rastru a vektoru. Každý z přístupů

[1] Uvedené souřadné systémy byly vybrány, jelikož jsou podstatné v následujících kapitolách práce.

má své uplatnění a obvykle jsou v GIS systémech kombinovány, což je patrné i z obrázku 2.2. Rastrové vrstvy obvykle používáme jako podkladové mapy při generování mapových výstupů pro zvýšení jejich informační hodnoty (letecký snímek, naskenovaná papírová mapa) nebo pro modelování spojitého jevu (výškový model terénu). Vektorové vrstvy obvykle klademe na podkladové rastrové vrstvy. Obsah vektorových vrstev představují aplikačně důležité prvky, které jsou v aplikaci spravovány a analyzovány.

Vektorové mapové vrstvy

Vektorové mapové vrstvy se zaměřují na jednotlivé geoobjekty v modelovaném prostoru. Prostorová poloha geoobjektů je reprezentována pomocí bodů, lomených čar a polygonů, tedy jednoduchých vektorových objektů [10, s. 17-18]. Při práci s vektorovými vrstvami jsou důležité jejich topologické vztahy [24, s. 29]. Obvykle zkoumáme především konektivitu (spojení dvojice geoobjektů), orientaci (určuje směr křivky), přilehlost (sousednost objektů) a obsažení (objekt je součástí jiného). Topologické vztahy ovlivňují rychlost analýzy dat a proto je často předpočítáváme.

Pro perzistentní uložení prostorových objektů vektorové mapové vrstvy se používá několik modelů s různou složitostí a mírou ukládaných informací spojených s topologií objektů:

- **Špagetový model** je nejjednodušším modelem. Všechny vektorové objekty jsou uloženy nezávisle na sobě. U každého objektu zaznamenáváme jeho typ (bod, lomená čára a polygon) a seznam souřadnic. Vzniká zde tedy redundance, jelikož mohou existovat body s duplikujícími se souřadnicemi, lomené čáry se společným nebo částečně společným průběhem atd. Jedná se o nejvýkonnější model, který ovšem kvůli absenci jakýchkoliv explicitně určených topologických informací je nevhodný pro analýzu dat (veškeré topologické vztahy objektů se počítají až před analýzou) [10, s. 19].

- **Základní topologický model** ukládá body a jejich spojnice (úsečky) do samostatných souborů. Složitější objekty jsou tvořeny z těchto jednoduchých objektů, na které se odkazují. Určování topologie objektů je zjednodušeno předpočítáním průsečíků objektů (bod průniku explicitně uložen do protínajících se objektů) a informaci o orientaci úsečky. Oproti špagetovému modelu snižuje redundanci a poskytuje lepší přístup pro analýzu dat. Nevýhoda modelu spočívá v chybějící uspořádanosti jednotlivých záznamů, což vede k časově neefektivnímu vyhledávání objektů v modelu [10, s. 20].

- **Hierarchický model** neukládá samostatně pouze body a čáry, ale i složitější objekty, které jsou hierarchicky tvořeny z jednodušších. Například polygon s dírami se odkazuje na uzavřené lomené čáry tvořící jeho vnější a vnitřní hranice. Každý objekt v modelu má předem známý typ a vyhledávání může probíhat nad menším množstvím dat, a tedy efektivněji. V hierarchickém modelu se nevyskytuje žádná redundance, ale je obtížně implementovatelný [10, s. 21].

Datová struktura modelu se na nižší vrstvě ukládá do souborů nebo databázových systémů. Soubory se obvykle používají pouze pro GIS aplikace vytvořené v generickém GIS nástroji (např. GRASS GIS[2]). Databázové systémy jsou častěji používány u robustních aplikací obsahujících kromě prostorových dat i mnoho neprostorových dat, které je obtížné ukládat pouze pomocí atributů geoobjektů. Databázové systémy také pomáhají zvyšovat

[2]GIS nástroj vyvinutý americkou armádou, který většinu operací provádí v příkazové řádce nad soubory uloženými v souborových systémech

otevřenost GIS aplikace díky snadnému přístupu k uloženým datům. Pro uložení prostorových dat je nutné v databázích zavést podporu pro prostorové datové typy a další nutné operace [24, s. 37–41]. Takto přizpůsobené databáze se nazývají *prostorové* a je jim věnována kapitola 3.

Rastrové mapové vrstvy

Rastrové mapové vrstvy se zaměřují na modelovaný prostor jako na celek. Modelovaný prostor je ve většině případech pravoúhlá výseč prostoru, která je pomyslně rozčleněna na dostatečně malé podčásti (buňky – obvykle čtvercové) nesoucí hodnotu (atribut) nějakého spojitého jevu [10, s. 24–25]. Vrstva je poté velmi podobná bitmapovým obrázkům. Podobnost není náhodná, jelikož jsou obrázkové soubory často použity jako datový nosič vrstvy.

Rastrová data jsou jednoduššího formátu než vektorová. Není nutné explicitně předpočítávat a ukládat topologické informace vektorů, jelikož jsou zřejmé z pozice buňky ve výseči prostoru. Díky pravidelnosti buněk postačuje při určování souřadnic pouze definovat souřadnice rohových buněk výseče [24, s. 32]. Oproti vektorové reprezentaci jsou hlavními nevýhodami vyšší paměťové nároky a nutnost předem určit neměnné rozlišení pro buňky vrstvy.

2.2 Vstupy a výstupy

Vstupy GIS rozlišujeme dle původu těchto dat na **primární zdroje dat** a **sekundární zdroje dat**. Primární zdroje dat jsou chápány jako měření dat v terénu, které je finančně i časově náročné. Je proto obvyklé poskytovat naměřená data jiným systémům. Tímto způsobem poskytnuté zdroje dat se nazývají sekundárními. Jedná se o data, která mohou pocházet například z digitalizovaných map a existujících geodatabází. Pro použití sekundárních dat je nutná jejich přesná definice (použitý souřadný sytém, kódování, přesnost měřicích přístrojů), která umožňuje, aby data byla v případě potřeby konvertována do podoby použitelné v dané GIS aplikaci [62, s. 115-118]. Možnosti a způsoby zprostředkování sekundárních zdrojů dat jsou popsány v kapitole 2.2.1.

Mapový výstup je nejdůležitějším a nejčastějším výstupem GIS systémů. Jedná se převážně o **tématické mapy**, které vznikly pomocí analýzy a následného proložení mapových vrstev. Mapový výstup je vzhledem podobný kartografickým mapám a taktéž je kompozicí stejných prvků (mapového pole, legendy, měřítka, názvu, vedlejší mapy atd.). Výhodou GIS oproti kartografickým mapám je schopnost vytvořit mapový výstup v reálném čase, což umožňuje využít mapový výstup pro okamžité zobrazení výsledků GIS operací [24, s. 68-75].

2.2.1 Způsoby výměny geodat

Pro sdílení sekundárních zdrojů dat se v počátcích GISů používalo jednoduché sdílení souborů, kdy datové soubory byly kopírovány mezi stanicemi. Tento přístup neumožňuje zajistit aktuálnost informací na jednotlivých stanicích jednoduchým způsobem ani zabránit kolizím při změně dat. Vylepšením popisovaného modelu je použití prostorových databází, které odstraňují obě nevýhody. Pokud chceme data poskytovat veřejně, není tento způsob ideální. Vedle bezpečnostního rizika se potýkáme s problémem otevřenosti systému, jelikož externí uživatel/nástroj před přístupem musí znát, jaká data se v databázi nachází a jak

jsou organizována. Problém lze vyřešit pomocí softwarové vrstvy, která poskytuje metadata o souborech a také může zajišťovat dodatečné služby, mezi které patří například autentizace a autorizace uživatelů. Je vhodné, aby softwarová vrstva pro snazší použití různými systémy byla standardizována. Pro snadnou distribuci informací jsou data umístěna na serverech. Server, který poskytuje geodata a obvykle používá pro jejich výměnu softwarovou vrstvu, se nazývá **mapový server**.

V průběhu vývoje GIS systémů vznikaly různé proprietární protokoly a způsoby pro výměnu dat, avšak byly vytlačeny standardizovanými řešeními, která umožňují snazší použití v různých systémech a nástrojích. Nejdůležitějšími standardy jsou WMS, WFS a WCS, které jsou detailně popsány ve zbytku této kapitoly.

Web Map Service (WMS)

Služba WMS [43] je uzpůsobena pro poskytování rastrových dat. Jedná se o nejpoužívanější formát pro sdílení geodat. Poskytovaná data není možné skrze WMS na serveru modifikovat. Komunikace probíhá ze strany klienta skrze HTTP protokol se specifikovanými parametry. Mapový server zpracuje požadavek a vrátí data dle typu příkazu. Rozlišujeme následující příkazy [51]:

- **GetCapabilities** – zjišťuje seznam poskytovaných služeb mapového serveru a jejich metadata. Klientovi se jako odpověď zasílá XML dokument ve formátu GML [40], který obsahuje informace o serveru a seznam poskytovaných vrstev a jejich parametrů (souřadný systém, ohraničení dat, slovní popis atd.).
- **GetMap** – získává rastrová data. Požadavek tvoří klient z informací získaných GetCapabilities příkazem a udáním požadované výseče vrstvy. Dostupné formáty, ve kterých jsou výsledná data odeslána ke klientovi, jsou taktéž určeny výsledkem GetCapabilies příkazu. Formát výsledných dat určí klient v zaslaném požadavku.
- **GetFeatureInfo** – poskytuje informace o objektu na specifikované pozici. Jedná se o nepovinnou část WMS standardu a ne všechny mapové servery jej podporují.

WMS je používána především pro sdílení podkladových mapových vrstev.

Web Feature Service (WFS)

Oproti službě WMS je služba WFS [42] uzpůsobena pro vektorová data včetně jejich modifikace. Komunikace probíhá pomocí protokolu SOAP [66], který přenáší GML dokumenty. Lze použít i jiné formáty (např. KML [41]).

Možnosti služby se zjišťují podobně jako v případě WMS příkazem GetCapabilities. Příkaz GetMap byl nahrazen příkazem **GetFeature**. Podpora pro modifikaci vektorových dat byla přidána ve verzi **Transactional WFS**, která obsahuje příkazy LockFeature k získání výlučného přístupu pro modifikaci geoobjektu a Transacion pro samotnou modifikaci dat. Transaction příkaz rozdělujeme na [51]:

- insertFeature – vložení nového geoobjektu,
- updateFeature – změnu existujícího geoobjektu,
- deleteFeature – odstranění existujícího geoobjektu.

WFS je použito všude tam, kde chceme poskytovat vektorová data, případně je umožňovat i vzdáleně modifikovat.

Web Coverage Service (WCS)

Služba WCS je podobná službě WFS, ale navíc umožňuje sdílení vícerozměrných dat měnících se v prostoru (např. data měnící se v čase) [51]. Je využívána spíše sporadicky, jelikož její vlastnosti jsou potřebné jen ve specifických případech.

2.3 Analýza dat

Analýza dat je jednou z nejdůležitějších funkcí GISu, která jej odlišuje od běžných map a mapových serverů. Mapy a mapové servery umožňují podobně jako GISy zobrazit prostorovou informaci ve formě map, ale neumožňují z dostupných informací tvořit nové. Analytických funkcí je celá řada, při pasportizaci počítačových sítí se můžeme setkat s následujícími kategoriemi analýz [24, s. 54-66]:

- **Prohledávání databáze** – vyhledáváme informace o geografické poloze, filtrujeme geoobjekty dle jejich atributů nebo provádíme topologické dotazy. Triviálním příkladem může být vyhledání optických vláken v maximální vzdálenosti 1 km od bytové jednotky s propustností nad 1 Gb/s.
- **Vzdálenostní analýzy** – zjišťují obalové zóny (zóna do určité vzdálenosti od geoobjektu) a určují nákladnost cesty. Tyto analýzy mají široké uplatnění při plánování různých projektů. Například při výstavbě nové optické trasy chceme zjistit, které obce se nachází do 5 km od trasy nebo pomocí dostupných geodat chceme určit nejvhodnější vedení nové trasy v krajině.
- **Analýzy modelů terénu** – vypočítávají sklony svahů a orientaci svahů z výškových modelů terénů. Využití lze nalézt při plánování výstavby rádiových spojů.
- **Analýzy sítí** – pracují s vektory modelující inženýrské a jiné sítě. Studují zatížení sítě a určují dopady výpadků v určitých částech sítě. Dále mohou být použity pro hledání trasy (směrování v počítačových sítích) a pro plánování výstavby nových částí sítě.

Důležitou vlastností GISů je, že analytické funkce je možné kombinovat a poté využívat pro různé organizace s rozdílnými cíli a požadavky.

Kapitola 3

Prostorové databázové systémy

Cílem prostorových (spatial) databázových systémů je umožnit efektivní dotazování nad uloženými daty, která zahrnují prostorovou složku. Vznik těchto systémů úzce souvisí s GIS systémy, které pro svou činnost potřebují pracovat s obrovským množstvím prostorových i neprostorových dat. Poté, co se v první generaci architektury GIS systémů ukázalo jako nevhodné ukládat data do souborů, objevily se první snahy o použití databází. Výsledkem bylo hybridní uložení dat, kde prostorová data byla nadále uložena v souborech a data neprostorová (atributy) byla přesunuta do *relačních databázových systémů*. Toto řešení nebylo na první pohled ideální, ale relační databázové systémy v té době neměly možnost prostorová data efektivně ukládat a pracovat s nimi. Původně bylo zamýšleno nahradit soubory a relační databázové systémy *objektovými databázemi*, jejichž principy vychází z objektově orientovaných jazyků (OOP) [52]. Rozšířenost relačních databází ovšem způsobila, že i přes inovativnost objektových databází pokračoval vývoj vylepšováním relačních databází, které přebíraly funkcionalitu dříve realizovanou na aplikační vrstvě. Rozšířením relačních databází o nové vlastnosti vznikly například objektově-relační, multimediální, temporální, aktivní a také prostorové databázové systémy. Souhrnně se nazývají *postrelační databázové systémy* a v současné době jsou nejpoužívanějšími databázovými systémy vůbec [28]. Prostorové databáze v GIS systémech sjednotily úložiště prostorových i neprostorových dat vektorového typu do jednoho databázového systému, což reflektuje současný trend vývoje GIS systémů. Rastrová data, jak již bylo ukázáno, se podobají spíše obrázkům, a proto jsou jim bližší multimediální databázové systémy. I když pro tento účel existují vhodné multimediální databáze, nejčastěji jsou rastrová data uložena v souborech a poskytována například skrze WMS. V prostorových databázích tedy ukládáme v součinnosti s GIS systémy pouze vektorová data.

Oproti běžným relačním databázím musí postrelační prostorové databáze navíc obsahovat podporu pro:

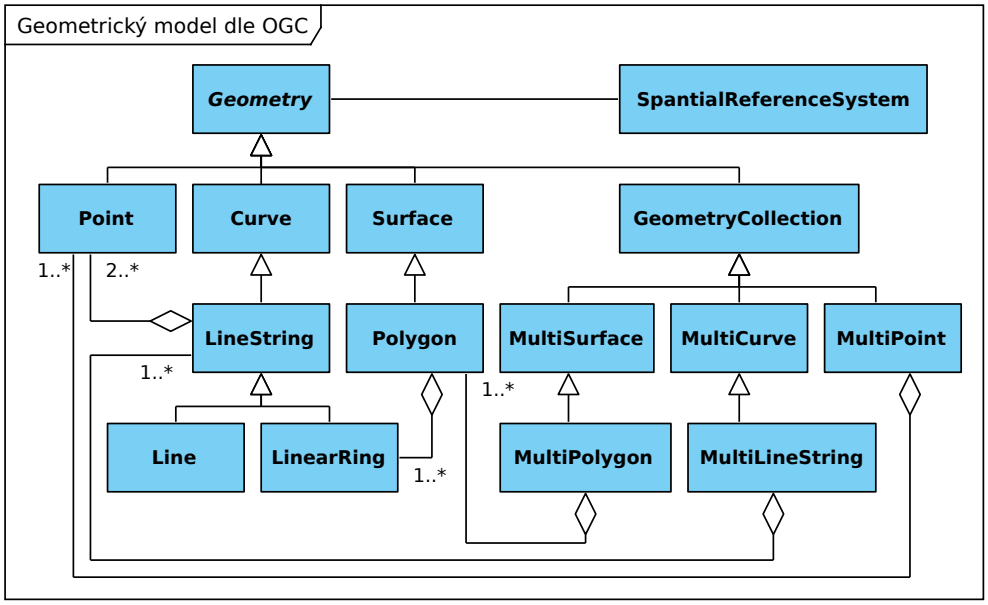
- prostorové datové typy – reprezentace bodů, křivek a povrchů,
- prostorové funkce – predikáty, relace a operace,
- indexování vícedimenzionálních dat.

3.1 Způsob reprezentace a uložení prostorových dat

Při uložení prostorových dat v prostorových databázových systémech lze navázat na principy definované v kapitole 2.1. Polohu geoobjektu vztaženou k jistému modelu prostoru

určujeme skupinou souřadnic definovaných v jistém souřadnicovém systému. Dle počtu a uspořádání souřadnic rozlišujeme geoobjekty na body, křivky a povrchy. Toto rozdělení je pouze základní a jednotlivé typy mohou být blíže specifikovány. Hierarchie typů geoobjektů se nazývá **geometrický objektový model**.

Geometrický objektový model definovaný organizací OGC[1] [44] a používaný v mírně pozměněných verzích v prostorových databázích je zobrazen na obrázku 3.1 ve formě diagramu tříd jazyka UML.



Obrázek 3.1: Model tříd geometrického modelu definovaného organizací OGC [44]

Geometrický objektový model může být implementován různými přístupy, které se obvykle liší dle výrobce databázového systému, a proto je nutné, aby byl model dostatečně abstraktní. Nejčastěji se setkáváme se třemi přístupy, které jsou specifikovány i organizací OGC [47]:

1. **Implementace pomocí numerických datových typů** – využívá samostatné databázové tabulky pro uložení jednotlivých typů geoobjektů. Geoobjekty jsou poté definovány záznamy v těchto tabulkách, na které se výsledné geoobjekty odkazují cizími numerickými klíči.

2. **Implementace pomocí binárních datových typů** – objekty jsou ze vstupních řetězců kódovány do binární podoby ve formátu *well-known binary (WKB) representation*. Uvedené řetězce jsou ve formátu *well-known text (WKT) representation*. WKT umožňuje jednoduše definovat všechny objekty modelu ve formě srozumitelné člověku (např. definice bodu lze pozorovat v kódu 3.1 na str. 15, řádek 3). WKT je používáno při dotazování nad geodaty pomocí SQL a při zpracování dotazu kódováno do WKB tvaru. Zakódovaný výstup může být proměnné velikosti.

3. **Implementace pomocí geometrických datových typů** – vychází z možnosti definovat nové datové typy dle standardu SQL 1999. Datové typy jsou vytvořeny dle modelu a jsou k nim přidány funkce.

[1] Open Geospatial Consortium – mezinárodní standardizační organizace působící na poli GIS

Nově vyvíjené prostorové databázové systémy jsou téměř výlučně implementovány pomocí třetího přístupu. Ostatní přístupy se naopak mohou vyskytovat u starších systémů. Každý přístup navíc používá meta tabulku, která pro každý sloupec s geometrickým typem v databázi obsahuje záznam s informací o použitém souřadném systému, typu geometrie atd. Tuto meta tabulku obsahují všechny pokročilé prostorové databáze.

3.2 Dotazování nad prostorovými daty a prostorové operace

Jelikož jsou prostorové databáze často postrelačního typu, pro dotazování se používá jazyk SQL. Je samozřejmě nutné, aby SQL dokázal pracovat s novými typy dat, definovanými v kapitole 3.1. Práce s datovými typy je odlišná dle způsobu jejich implementace [47].

U implementací pomocí numerických a binárních datových typů se zadání a vypsání hodnoty geografického typu provádí pomocí WKB a WKT formátů, jejichž vzájemný převod je obstaráván databázovými funkcemi. Demostrace takového přístupu použitelného v MySQL databázovém systému [56] je dostupná v kódu 3.1. Kód obsahuje SQL skript, který na řádku 1 definuje schéma tabulky se sloupcem s prostorovým datovým typem. Na řádku 2 je nad sloupcem vytvářen prostorový index. Řádek 3 realizuje vložení záznamu do tabulky. Vkládaný bod je zadán ve WKT formátu. Poslední řádek příkladu obsahuje příkaz pro získání všech bodů z tabulky ve formátu WKB a WKT.

Prostorové databáze implementované pomocí geometrických datových typů mohou použít výše uvedený přístup. Navíc lze zadávat hodnotu typu pomocí konstruktoru (např. Oracle Spatial [29]).

```
1 CREATE TABLE geom (g GEOMETRY);
2 ALTER TABLE geom ADD SPATIAL INDEX(g);
3 INSERT INTO geom VALUES (GeomFromText('POINT(1 1)'));
4 SELECT AsBinary(g) AS gb, AsText(g) AS gt FROM geom WHERE Dimension(g) = 0;
```

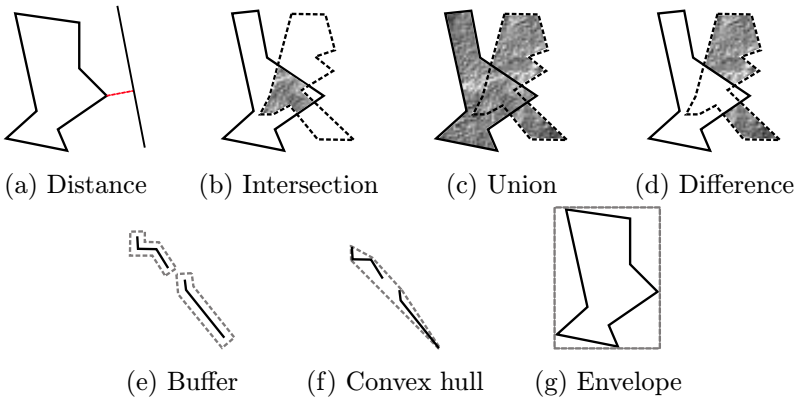
Kód 3.1: Příklad SQL dotazů pracujících s prostorovým datovým typem v MySQL

Jak je z kódu 3.1 patrné, pro práci s datovými typy se používají operace, které jsou v SQL reprezentovány funkcemi. Aby se operace nelišily u každého systému, standard OGC definující geometrický model specifikuje také jednotlivé operace použitelné nad jednotlivými objekty modelu. Tento standard přijali až na výjimky téměř všichni výrobci prostorových databází.[2] Operace lze rozdělit do následujících kategorií [44]:

- **prostorové binární predikáty** – testují vztah mezi dvěma geoobjekty (`Equals` – shoda, `Disjoint` – prázdný průnik, `Touches` – dotek, `Within` – je uvnitř, `Contains` – je obsažen, `Overlaps` – částečný překryv, `Crosses` – kříží se, `Intersects` – mají průnik).
- **prostorové operátory** – jejich návratem je nový geoobjekt (`Intersection` – průnik s jiným geoobjektem, `Union` – sjednocení s jiným geoobjektem, `Difference` – rozdíl s jiným geoobjektem, `Buffer` – okolí geoobjektu v určité vzdálenosti, `ConvexHull` – konvexní obálka geoobjektu, `Envelope` – *bounding box* kolem geoobjektu). Příklady jednotlivých operátorů jsou ukázány na obrázcích 3.2.

[2] Můžeme se ovšem setkat s úpravou pojmenování operací z důvodů zabránění kolizí. Například jména prostorových funkcí a prostorových datových typů v Oracle Spatial prostorové databázi jsou uvozeny prefixem `SDO_`.

- **ostatní** – jedná se o operace, které jsou speciální nebo se vztahují jen k některým geoobjektům. (`Distance` – nejkratší vzdálenost dvou geoobjektů, `Dimension` – dimenze geoobjektu daná jeho typem, `AsText` – převod do WKT formátu, `AsBinary` – převod do WKB formátu, `X` – x souřadnice bodu u objektu `Point` atd.)



(a) Distance (b) Intersection (c) Union (d) Difference

(e) Buffer (f) Convex hull (g) Envelope

Obrázek 3.2: Příklady prostorových operátorů

Operace z kategorie predikátů jsou důležité například při práci s GIS, který používá špagetový datový model. Při použití špagetového modelu se totiž pomocí prostorových predikátů počítají veškeré topologické vztahy mezi geoobjekty. Prostorové operátory nachází uplatnění v GIS systémech při analýze dat.

Kapitola 4

Pasportizace počítačových sítí

Pasportizace je proces, jehož předmětem je inventarizace stavu, konstrukce a prvků reálných objektů do technické dokumentace zvané *pasport*. Nejčastěji se můžeme s pasportizací a pasporty setkat v oboru stavebnictví, jelikož od počátku civilizace se projevovala nutnost vedení stavu a evidence nemovitostí. Typickým pasportem v oblasti nemovitostí je složka obsahující potřebné výkresy a technické zprávy. Dříve byl pasport uchováván v papírové podobě. V dnešní době je nejčastěji uchováván elektronicky. U odvětví, jako je stavebnictví, vytváříme elektronický pasport takovým způsobem, aby korespondoval s existujícím pasportem. Tento přístup lze pozorovat například ve vztahu papírových výkresů a elektronických výkresů vypracovaných pomocí CAD softwarových nástrojů. U odvětví, ve kterých neexistuje legislativní ani zažitý způsob pasportizace, se zabýváme návrhem vhodného nového formátu a obsahu elektronického pasportu s cílem dosažení nejefektivnějšího pasportizačního procesu. Efektivita procesu pasportizace je důležitá především z ekonomického aspektu, protože při změnách reálných objektů (např. rekonstrukce domu) je nutné pasportizaci opakovat a tedy znovu vynaložit náklady na její provedení [11].

Pasportizace počítačových sítí nemá pevný základ v legislativě ani neexistuje převládající forma pasportu. Existuje mnoho přístupů pro pasportizaci počítačových sítí. Některé volí cestu připodobnění inženýrským sítím z oboru stavebnictví. Uvedený přístup umožňuje popsat fyzickou strukturu sítě a polohu jejich jednotlivých prvků pomocí pasportizačních metod a nástrojů zavedených ve stavebnictví, ale jen obtížně umožňuje evidovat její logickou strukturu (evidence IP adres, podsítí atd.). Další používaný přístup se zaměřuje na pasportizaci logické struktury, ale fyzickou strukturu dokumentuje jen povrchně nebo vůbec [30]. Tento přístup může být vhodný pro metalické a bezdrátové počítačové sítě, ale nevyhovuje u rozsáhlých optických sítí, kde je nutné provázat logickou a fyzickou strukturu do co největšího detailu. Při pasportizaci optických počítačových sítí se často setkáváme s použitím obou přístupů zároveň. Kombinované řešení umožňuje přesnou pasportizaci, ale její evidence a správa je náročná, jelikož se jedná o dva nezávislé pasportizační procesy.

Zbytek této kapitoly si klade za cíl shromáždit informace o pasportizovaných objektech (prvky sítě, jejich vzájemné propojení a způsob výstavby) a posléze na základě těchto informací definovat vhodný formát pro elektronický pasport. Z omezeného rozsahu dokumentu plyne, že není možné uvést a detailně popsat princip všech pasportizovaných objektů z oboru počítačových sítí. Cílem kapitoly je shromáždit a kategorizovat stěžejní prvky sítí společně s nejdůležitějšími principy a postupy aplikovanými při výstavbě síťové infrastruktury. Výsledný formát pasportu se také drží této filozofie.

4.1 Prvky sítí

Existuje nepřeberné množství prvků počítačové sítě. Pro lepší přehled budeme v tomto dokumentu rozlišovat síťové prvky na **aktivní** a **pasivní**. Terminologie je převzata z oboru elektrotechniky, kde jako aktivní označujeme elektronickou součástku, která není schopna pracovat bez napájení a pasivní jako součástku, která naopak ke své funkci napájení nepotřebuje [57]. Tato kategorizace byla zvolena, jelikož se mimo jiné běžně používá k rozlišení prvků optických sítí.

4.1.1 Aktivní síťové prvky

Mezi aktivní síťové prvky se řadí zesilovače (amplifier), opakovače (repeater), rozbočovače (hub), můstky (bridge), přepínače (switch), směrovače (router), brány (gateway) a síťové karty [60]. Obvykle slouží pro spojení sítí do větších celků. Z pohledu pasportizace fyzické struktury počítačové sítě se u aktivních síťových prvků kromě obecných parametrů (výrobce, typ, napájení atp.) zajímáme především o počet a typ instalovaných portů.

4.1.2 Pasivní síťové prvky

Pasivní prvky se často odlišují dle použitého přenosového média. Běžně se používají tři typy přenosových médií s následujícími prvky:

- **Metalické spoje** – obsahují zejména následující pasivní prvky: kabely, konektory, propojky, rozdvojky a patch panely [60]. Jsou obvykle používány v LAN sítích nebo v sítích postavených na technologii DSL [19]. Z pohledu pasportizace počítačových sítí nejsou tak podstatné jako optické spoje, ale jejich evidence je nutná, jelikož širokopásmová optická linka nevede obvykle až ke koncové službě (do bytové jednotky), ale v určitém bodě je nahrazena metalickým vedením.
- **Optické spoje** – sestávají z pasivních prvků, jako jsou: optická vlákna, optické kabely pro shlukování vláken, mikrokabely pro techniku mikrotrubičkování [5], konektory, spojky, kazety pro umístění optických svárů, rozvaděče, pasivní rozbočovače (splitter) a další prvky spojené s optoelektronikou [14].
- **Rádiové spoje** – jsou tvořeny anténami, které jsou pro tyto spoje dominantními pasivními síťovými prvky [60].

4.2 Způsob výstavby

Způsob výstavby sítí se odlišuje použitými spoji, typy a výrobci aktivních prvků, topologiemi apod. Obecně lze počítačové sítě rozdělit na **páteřní** – zajišťují spojení mezi uzly sítě, **přístupové** – umožňují přenos mezi síťovými uzly a koncovými terminály a **lokální** – nahrazují koncový terminál za geograficky malou síť budovanou většinou koncovými uživateli. Pro pasportizaci jsou podstatné především sítě páteřní a přístupové, které jsou v současné době budovány pomocí optických sítí. Při pasportizaci optických sítí je podstatná realizace celé optické trasy, a proto je nutné uvést její dílčí součásti.

4.2.1 Technická realizace optické trasy

Při realizaci optické trasy je důležitý především způsob umístění optických kabelů do optické trasy a technologie spojování optických kabelů. Optické trasy jsou umísťovány do [5]:

- Výkopu – optický kabel s vhodně uzpůsobeným pláštěm je zakopán do výkopu bez ochranného prvku. Alternativou k zemním výkopům je pokládání mikrokabelů do drážek ve vozovkách, chodnících apod. Optické kabely ve výkopech bývají překryty výstražnými fóliemi.
- Vzduchu – optický kabel s tažným prvkem je veden vzduchem. Na delší vzdálenosti je nutné použít, podobně jako u elektrického vedení, sloupy.
- Kabelovodu – optické kabely jsou ukládány do existujících kabelovodů. Optický kabel může být umístěn přímo do chráničky. Častěji se ale setkáváme s chráničkami obsahujícími několik *mikrotrubiček (microduct)*. Mikrotrubička je menší chránička pro vložení optických kabelů. Její výhoda spočívá v možnosti zavedení mikrokabelu do již položených chrániček pomocí *mikrotrubičkování*.
- Kanalizace atd.

Pro budování dlouhých tras nebo například při rozbočení pomocí pasivního rozbočovače je nutné kabely spojovat. Spojování probíhá na úrovni vláken kabelu. Podobně jako u metalického vedení lze pro spojení použít konektory. Konektory však mají velký útlum, a proto se častěji používají nerozebiratelná spojení vláken ve formě svařeného spoje. Optické sváry je vhodné zajistit před mechanickým poškozením v optické vaně [5].

4.3 Formát pasportu

Formát pasportu pro pasportizaci fyzické struktury počítačové sítě musí předepisovat typy pasportizovaných prvků sítě, způsob, jakým mohou být prvky navzájem propojeny, sledované parametry prvků a způsob identifikace geografické polohy prvků. Nesledujeme pouze síťové prvky, ale také prvky použité v trasách počítačových sítí a v některých případech také samotné trasy.

4.3.1 Sledované prvky

Tabulka 4.1 na straně 20 obsahuje seznam pasportizovaných typů prvků sítě a tras včetně jejich parametrů a vzájemného propojení. Všechny typy uvedené v tabulce mají s výjimkou typu „ostatní“ základ v prvcích sítě a prvcích tras představených v této kapitole. Typ ostatní je abstraktní typ, kterým lze evidovat všechny aktivní síťové prvky a některé pasivní síťové prvky (antény). Specifika různých typů prvků jsou v zařízení modelována pomocí parametrů odlišných pro každý typ.

4.3.2 Geografická poloha sledovaných prvků

Pasportizace fyzické struktury počítačových sítí a jejich tras vyžaduje u jednotlivých prvků zaznamenávat přesnou geografickou polohu. Z pohledu geografické polohy rozlišujeme prvky, které mají:

- **bezrozměrnou polohu** (ve formě bodu) – sloup, optická vana, optický rozvaděč, optický svár, splitter, koncovka, port a zařízení.
- **jednorozměrnou polohu** (ve formě křivky) – výkop, výstražná fólie, chránička, mikrotrubička, kabel a optické vlákno.

Vícerozměrnost jednotlivých prvků je zanedbávána, jelikož by u takto reprezentovaných prvků bylo obtížné určovat vzájemnou topologii prvků. Vícerozměrnost může být ovšem do jisté míry evidována pomocí parametrů (např. u chráničky evidujeme její průměr).

Pro kategorizaci částí sítě je vhodné kromě geografické polohy pasportizovat u významných prvků také poštovní adresy.

Tabulka 4.1: Tabulka typů síťových prvků s parametry a vzájemným propojením

Název	**Parametry**	**Propojení**
Sloup	výška uložení kabelu	obsahuje kabely
Trasa	typ (výkop, kanalizace atd.)	obsahuje výstražné fólie, chráničky, mikrotrubičky a kabely
Výstražná fólie	hloubka uložení, typ[1] (výrobní název, šířka, barva)	uložena ve výkopu
Chránička	hloubka uložení, délka, barva, typ[1] (výrobní název, vnitřní a vnější průměr)	uložena ve výkopu; obsahuje mikrotrubičky a kabely
Mikrotrubička	délka, barva, typ[1] (výrobní název, vnitřní a vnější průměr)	uložena ve výkopu
Kabel	délka, barva, typ[1] (výrobní název, typ přenosového média)	uložen ve výkopu, na sloupech nebo samostatně; metalický kabel je zakončen koncovkami, optický může obsahovat optické buffery a optická vlákna
Optický buffer	barva, typ[1] (počet obsažených optických vláken)	součást optického kabelu; obsahuje optická vlákna
Optické vlákno	barva, typ[1] (název, typ)	součást optického kabelu nebo buffer; ukončeno koncovkami nebo je konec svařen s jiným vláknem
Optický svár	pozice v optické vaně, útlum	spojuje dva konce optických vláken; umístěn v optické vaně
Optická vana	typ[1] (výrobní název, maximální počet obsažených svárů)	součást optického rozvaděče; obsahuje optické sváry
Optický rozvaděč	popis, typ[1] (výrobní název, typ, počet optických van)	obsahuje optické vany a zakončení optických vláken
Splitter	popis, typ[1] (výrobní název, poměr rozdělení, útlum, typ, dle typu může obsahovat délku vláken a typy konektorů vláken)	rozděluje jedno optické vlákno (master) na více vláken (slaves) dle poměru rozdělení
Koncovka	typ[1] (výrobní název, typ přenosového média)	ukončuje metalické kabely nebo optická vlákna; připojuje se do portu
Port (zásuvka)	typ[1] (výrobní název, typ přenosového média)	obsažen v zařízení; je k němu připojen konektor
Ostatní	popis, typ[1] (výrobní název, typ, typy parametrů), parametry[2]	obsahuje porty

[1] Jedná se o typ pro kategorizaci daného prvku obsahující parametry uvedené v závorce.
[2] Specifické parametry pro jednotlivé prvky typu „ostatní“

4.3.3 Softwarový pasport

Pasport s navrženým formátem lze uložit do elektronické databáze, což je ukázáno v kapitole 5.2.3. Takto uložená data pasportu lze použít jako datový základ pro informační systém, jehož funkcí je vizualizace evidovaných dat, správa evidovaných dat a poskytování evidovaných dat dalším systémům. Pasport obsahuje geografická data, která lze vizualizovat do grafické podoby ve formě map. Pro efektivní správu dat je nutné, aby informační systém umožňoval kromě vizualizace geografických poloh pasportizovaných prvků také jejich vkládání a modifikaci. Informační systém s takovou funkcionalitou lze zařadit mezi geografické informační systémy. Výsledný softwarový pasport je realizovatelný geografickým informačním systémem napojeným na elektronickou databázi, která ukládá data jak prostorového, tak i neprostorového charakteru. Následující kapitoly se budou věnovat analýze, návrhu a implementaci takového softwarového pasportu.

Kapitola 5

Analýza a návrh

Analýza a návrh geografického informačního systému pro pasportizaci počítačových sítí navazuje na kapitolu 4.3, která předepisuje požadavky na pasportizaci a softwarový pasport. V návaznosti na uvedené definice se v této kapitole budou dále rozvíjet a analyzovat definované požadavky a také budou ukázány existující informační systémy, které se zabývají podobnými tématy. V druhé části kapitoly bude navržen informační systém splňující představené požadavky.

5.1 Analýza

5.1.1 Specifikace požadavků

Požadavky na systém lze rozdělit na hlavní, které jsou spojeny s primárními funkcemi a účelem sytému a vedlejší, které doplňují funkce přímo nesouvisející s hlavním účelem systému, ale jsou nezbytné pro jeho správnou činnost a předpokládanou funkcionalitu.

Hlavní požadavky

- možnost dokumentovat a vizualizovat fyzickou strukturu počítačových sítí včetně geografické polohy prvků sítě a návaznosti prvků (formou interaktivní mapy);
- možnost dokumentovat a vizualizovat podstatné části tras pro vedení počítačových sítí (formou interaktivní mapy);
- vizualizace hierarchického zanoření prvků v počítačové síti a jejich trasách;
- možnost budoucí realizace/napojení na systém pro dokumentaci logické struktury počítačové sítě (podsítě, VLAN sítě atd.).

Výčet prvků počítačové sítě a tras, které systém musí být schopen dokumentovat a vizualizovat, je shodný s prvky z tabulky 4.1 na straně 20.

Vedlejší požadavky

- autorizace a autentizace uživatelů s hierarchií přístupových práv;
- multiplatformnost – nutná podpora přístupu uživatele k systému z operačních systémů Microsoft Windows, Linux a OS X;

- intuitivní uživatelské prostředí s možností nastavení chování dle uživatelských preferencí;
- internacionalizované uživatelské prostředí (angličtina a čeština s možnostmi doplnění dalších jazyků);
- pasportizace počítačové sítě simulující způsob její reálné výstavby z důvodu rychlé orientace nového uživatele;
- centralizovaný přístup;
- možnost interakce s katastrálními mapami České republiky;
- vyhledávání a označování poštovních adres na mapě;
- import dat z existujících zdrojů (výkresy vytvořené v CAD nástrojích);
- použití otevřených nebo svobodných technologií při implementaci;
- nízké nebo žádné pořizovací náklady na využité komponenty třetích stran;
- možnost budoucího rozšíření komponent systému.

5.1.2 Analýza požadavků

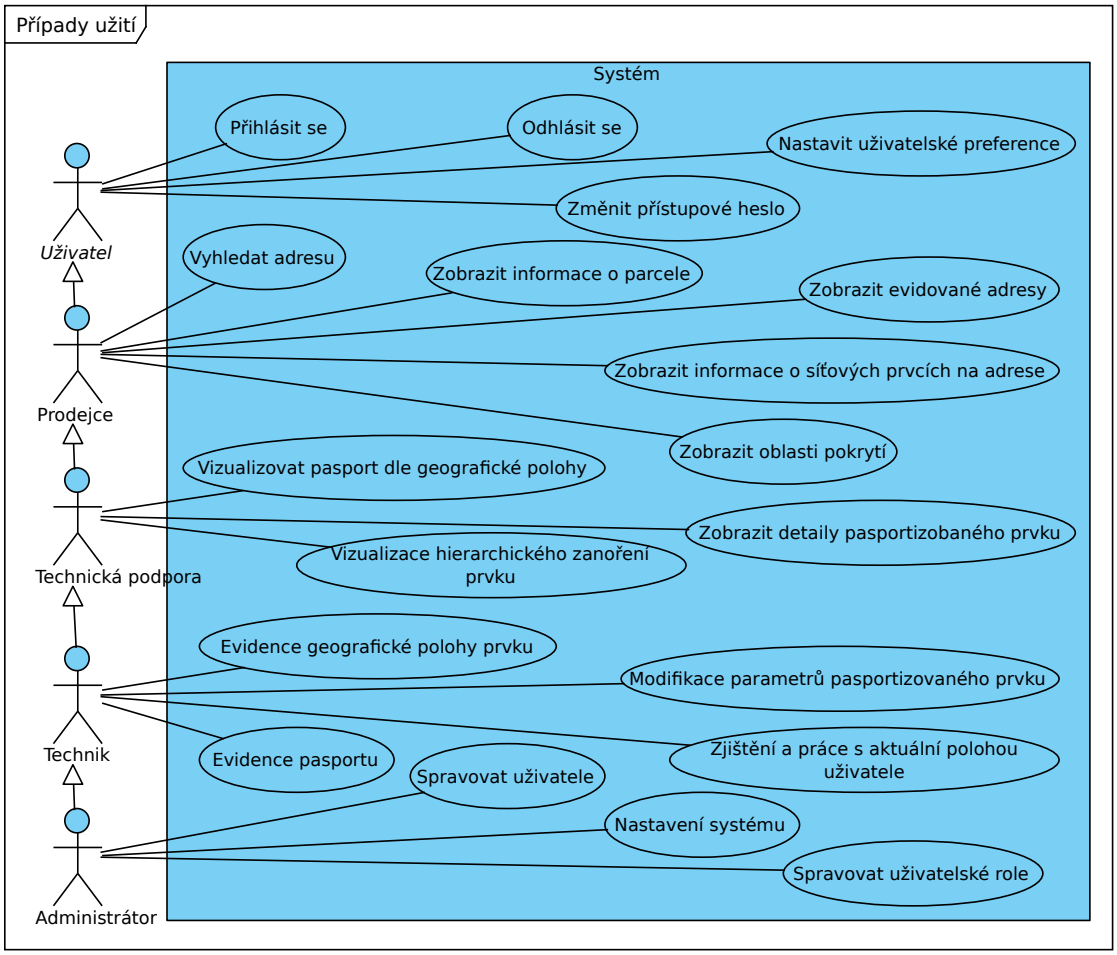
V systému se bude vyskytovat několik rolí s různými přístupovými právy:

- **Prodejce** – je mu umožněno pouze prohlížet základní údaje spojené s dostupností sítě v adresních bodech a geografických lokacích. Struktura sítě není pro tohoto uživatele podstatná a může být vhodně potlačena pro snazší práci uživatele.
- **Technická podpora** – disponuje právy pro vizualizaci všech částí systému, ale pouze pro čtení.
- **Technik** – zahrnuje práva technické podpory, navíc je schopen modifikovat evidovaná data spojená s pasportizací sítě. Uživatelé v roli technika se často pohybují v terénu, proto systém musí pro tyto účely zjednodušovat pasportizaci dle aktuální pozice technika.
- **Administrátor** – disponuje právy technika, spravuje uživatelské účty a modifikuje nastavení systému. Administrátor může v organizaci plnit roli technika, a proto má navíc přistup ke stejným funkcím jako technik.

Nezávisle na uživatelské roli je uživatel schopen spravovat svůj uživatelský profil a preference. Všechny funkce z pohledu uživatele reflektující jednotlivé uživatelské role jsou k dipozici v diagramu případů užití na obrázku 5.1.

5.1.3 Existující řešení

Pasportizace počítačových sítí není novým oborem, a proto existuje několik hotových řešení. Najít vyhovující řešení, které splňuje všechny definované požadavky bez omezení, je ovšem obtížné. Následující text popisuje základní charakteristiky vybraných řešení společně s rozborem jejich nevhodnosti pro splnění definovaných požadavků. Pro jednotlivá řešení jsou vypsány nevýhody, které měly za příčinu nepoužití tohoto řešení a vývoj specifikovaného systému. Nevýhody jsou ve výčtu, jehož jednotlivé položky jsou uvozeny symbolem ✗.



Obrázek 5.1: Případy užití systému reflektující jednotlivé uživatelské role ve formě diagramu případů užití jazyka UML

cableScout

cableScout je software, jehož prioritním cílem je dokumentace počítačových a telekomunikačních sítí se snadným napojením na GIS, systémy pro monitorování a správu sítě s možností doplnění funkcionality pomocí modulární architektury [25].

- ✗ neumožňuje evidenci rádiových spojů,
- ✗ je dostupný pouze pro platformu Microsoft Windows,
- ✗ dodavatel dodává pouze řešení na míru odběrateli (neuvádí cenu),
- ✗ jako úložiště dat používá databázi Oracle s vysokými pořizovacími náklady,
- ✗ chybí internacionalizace programu do českého jazyka.

OSP InSight

OSP InSight je software pro dokumentaci sítí založený na GIS řešeních jiných společností, ze kterých přebírá i uživatelské rozhraní [1].

- ✗ nelze evidovat trasy,

- ✗ nelze vizualizovat hierarchické zanoření prvků v počítačové síti a jejich trasách,
- ✗ neumožňuje evidovat logickou strukturu sítě,
- ✗ je dostupný pouze pro platformu Microsoft Windows (částečně pro ostatní platformy),
- ✗ dodavatel dodává pouze řešení na míru odběrateli,
- ✗ grafické uživatelské rozhraní je neintuitivní,
- ✗ chybí internacionalizace programu do českého jazyka.

NetDoc2

NetDoc2 je software ve formě webové aplikace pro dokumentaci sítí [8].

- ✗ neumožňuje evidovat geografickou polohu prvků,
- ✗ disponuje omezenými možnostmi pro evidenci optických sítí,
- ✗ neumožňuje evidovat logickou strukturu sítě,
- ✗ chybí internacionalizace programu do českého jazyka.

SPIDER-Fiber

SPIDER-Fiber je robustní software pro tvorbu a správu výkresové dokumentace a projektování optických kabelových sítí, který je založen na technologiích společnosti Bentley (MicroStation) [18].

- ✗ vhodný pouze pro evidenci optických sítí,
- ✗ založen na CAD systému – obtížná správa geografické polohy a logické struktury sítě,
- ✗ dostupný pouze pro platformu Microsoft Windows,
- ✗ vysoké pořizovací náklady.

5.2 Návrh

Ideálním médiem pro vizualizaci pasportu zahrnujícího prvky s definovanými geografickými polohami je interaktivní mapa, jejíž součástí je mimo jiné možnost vizualizace detailů jednotlivých pasportizovaných prvků. Mapa přináší mnohé výhody, mezi které patří snadná lokalizace prvků, rychlá orientace, možnost použití podkladové mapy pro zvýšení informační hodnoty (např. ortofotomapy či mapy katastru nemovitostí) atd. Přehledná mapa musí vhodně volit symboly, kterými zobrazuje jednotlivé prvky. V opačném případě by bylo obtížné od sebe rozeznat například splitter a optický rozvaděč. Volba symbolů ovšem neposkytuje takovou možnost u prvků s geografickou polohou ve formě křivek (kabel, výkop atd.). U křivek máme možnost ovlivnit pouze jejich šířku, barvu či typ, a proto není vhodné zobrazovat všechny prvky na mapě zároveň, ale vybrat pouze některé typy. Tématické rozdělení prvků vede k použití mapových vrstev, které mohou být zobrazovány dle požadavků uživatele na výstupní mapu. Mapové vrstvy lze realizovat pomocí GIS systémů s mapovým výstupem. U obecných GIS systémů se setkáváme pouze s omezenou možností vizualizace

neprostorových dat (viz nevýhody OSP In Sight v kapitole 5.1.3). Obecné GIS systémy tedy nelze bez úprav využít, proto je nutné navrhnout vlastní GIS systém s možností pokročilejší vizualizace a správy neprostorových dat. Tato kapitola se zabývá návrhem právě takového systému.

5.2.1 Architektura systému

Bylo navrženo, aby GIS systém byl realizován jako webově orientovaná aplikace. Realizací ve formě webové aplikace je implicitně vyřešen požadavek na centralizovaný přístup k systému. Podstatný je také fakt, že vývoj jedné komplexní webové aplikace je méně časově náročný v porovnání s vývojem klientských aplikací pro všechny podporované operační systémy.

Realizace systému pomocí webové aplikace byla umožněna díky existenci knihovny OpenLayers, která slouží k vytváření interaktivních map v prostředí webového prohlížeče [22]. Existuje několik podobných knihoven a nástrojů (např. Google Maps API, Mapy.cz API), ale žádný nedosahuje možností knihovny OpenLayers. Mezi hlavní přednosti OpenLayers patří podpora zobrazení prostorových dat z různých zdrojů (WMS, WFS, Bing Maps, OpenStreetMaps atd.), podpora mapových projekcí, rozšiřitelná paleta ovládacích prvků mapy, vestavěná funkcionalita pro modifikaci vektorových geoobjektů, otevřené zdrojové kódy, rozsáhlá dokumentace atd. Knihovna a definice pro tvorbu interaktivní mapy jsou vkládány do stránek webové aplikace a při každém načtení takové stránky je mapa vykreslena. Inicializace mapy je poměrně zdlouhavý a náročný proces, a proto je vhodné inicializovat mapu pouze při prvním požadavku uživatele a následujícími požadavky pouze dynamicky měnit obsah stránky s vykreslenou mapou. Pro zamezení opětovného načítání stránek webové aplikace lze použít technologii AJAX, která umožňuje asynchronní komunikaci se zdroji dat [23]. V knihovně OpenLayers probíhá veškerá komunikace za účelem získání prostorových dat právě pomocí technologie AJAX. Ostatní části webové aplikace se tomuto faktu musí přizpůsobit – pro komunikaci se zdroji dat musí používat výlučně technologii AJAX.

Webovou aplikaci s persistentním úložištěm dat, která je budovaná technologií AJAX, je vhodné rozdělit na část zodpovědnou za generování/poskytování stránek webové aplikace, část zodpovědnou za poskytovaní dat a část zodpovědnou za perzistentní uložení dat. Vygenerované stránky obsahují klientské skripty, které dle uživatelských akcí asynchronně komunikují s částí určenou pro poskytování dat za účelem získání a modifikace dat. Zdrojem perzistentně uložených dat je databáze. Zvýšení transparentnosti komunikace mezi klientskými skripty a částí poskytující data lze docílit použitím softwarového rozhraní společně s vhodným formátem pro přenášená data. V navrhovaném systému není uvedený postup rozdělení webové aplikace dostačující, jelikož pro poskytování prostorových a neprostorových dat se používají rozličné přístupy. V souvislosti s prostorovými daty se dle jejich typu používá WMS či WFS služba, kterou lze vytvářet pomocí mapového serveru napojeného na zdroj dat. Zdrojem prostorových dat může být databáze s prostorovými daty, kolekce souborů či externí mapový server. Do návrhu systému byl proto přidán mapový server. Nutnost práce s prostorovými daty vedla k nahrazení obecné databáze za databázi prostorovou. Návrh architektury systému se drží popsaného rozdělení a obsahuje následující části:

- **nd-client** – generuje stránky/fragmenty stránek webové aplikace, které tvoří uživatelské rozhraní systému. Hlavním prvkem uživatelského rozhraní je interaktivní mapa pro zobrazení prostorových dat. Zobrazovaná prostorová data jsou získávána asynchronně z WMS a WFS služeb umístěných na lokálním mapovém serveru a externích mapových serverech. Mapa je realizována pomocí knihovny OpenLayers. Mimo interaktivní mapu vizualizuje část nd-client neprostorová data, která jsou získávána

asynchronně z části nd-rest-backend. Uživatelské rozhraní obsažené v části nd-client neumožňuje data pouze vizualizovat, ale také vytvářet nová a modifikovat stávající. Zadání nových a upravených prostorových dat probíhá kreslícími nástroji interaktivní mapy. Neprostorová data jsou zadávána pomocí kolekce formulářů.

- **nd-rest-backend** – spravuje neprostorová data uložená v prostorové databázi a poskytuje rozhraní k jejich získání a modifikaci. Rozhraní je koncipováno architekturou rozhraní *REST* [64]. Data přenášená mezi částmi nd-client a nd-rest-backend jsou strukturovaná do objektů. Pro přenos objektů byl zvolen formát JSON. Jelikož jsou data, která část nd-rest-backend poskytuje, perzistentně uložena v postrelační databázi, je nutné převádět záznamy jednotlivých tabulek databáze na objekty. Převod lze řešit technikou *objektově-relačního mapování (ORM)* [55].

- **Lokální mapový server** – poskytuje prostorová data uložená v prostorové databázi skrze WFS službu ve formě tématických vektorových mapových vrstev. Vektorové mapové vrstvy obsažené v systému lze v kontextu GIS považovat za primární data. Lokální mapový server slouží navíc jako vyrovnávací paměť pro zprostředkování rastrových mapových vrstev umístěných na nespolehlivých externích mapových serverech.

- **Externí mapové servery** – poskytují rastrové podkladové mapové vrstvy skrze WMS nebo jinou službu. Jedná se například o Google Maps, Bing Maps nebo katastrální mapy České republiky. Data poskytovaná externími mapovými servery lze v kontextů GIS považovat za sekundární. Podkladové mapové vrstvy z nespolehlivých externích mapových serverů mohou být zprostředkovávány lokálním mapovým serverem.

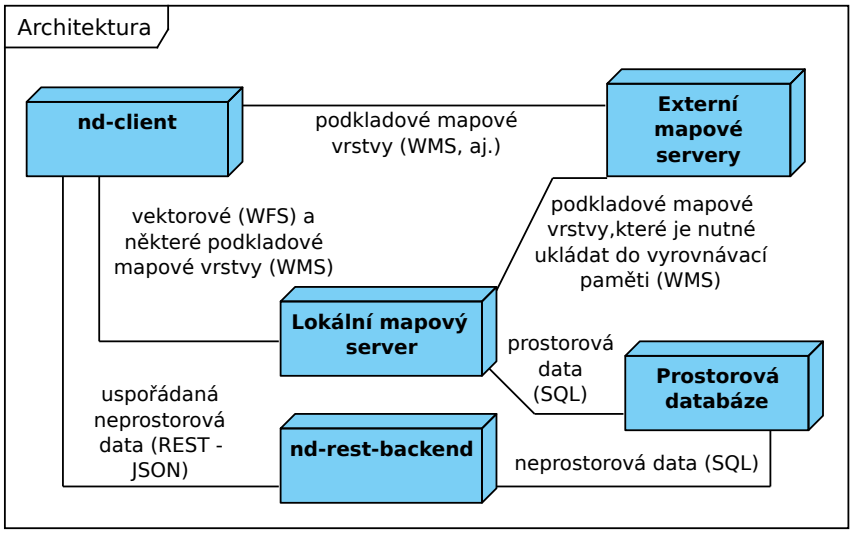
- **Prostorová databáze** – jedná se o postrelační databázový systém odpovídající definici z kapitoly 3, který obsahuje data spojená s pasportizací a data nutná pro správu uživatelských účtů. K databázi přistupují části nd-rest-backend a lokální mapový server, které se dotazují nad daty v databázi pomocí jazyka SQL. Inicializace schématu databáze je řízena částí nd-rest-backend.

Navrhovanou architekturu systému včetně vzájemných vazeb jednotlivých podčástí systému lze pozorovat na obrázku 5.2 na straně 28.

V případě lokálního mapového serveru a prostorové databáze lze použít hotová řešení, a proto se jimi návrh nebude dále zabývat. Části nd-client a nd-rest-backend jsou závislé na datovém modelu, a proto nelze použít žádné existující generické řešení a je nutné se věnovat jejich návrhu.

nd-rest-backend

Část nd-rest-backend spravuje data v prostorové databázi a ve formě objektů je poskytuje jako dostupné zdroje na vstupně/výstupní rozhraní založeném na architektuře REST. Úkolem části nd-rest-backend je přijmout přes komunikační rozhraní požadavek na získání/modifikaci dat, transformovat data zaslaná v požadavku na objekty, získat/modifikovat požadovaná data z/v databázi, transformovat získaná/modifikovaná data do objektů a zaslat je jako odpověď na přijatý požadavek. Uvedený popis nezahrnuje složitější operace, kdy je pro vykonání operace nutná komplikovanější aplikační logika spojená se získáním/modifikací dat. Z popisu lze ale vypozorovat, že je možné rozdělit dílčí úkoly do tří skupin:



Obrázek 5.2: Komponenty (části) navrženého GIS systému s uvedenými vzájemnými vztahy

- Komunikace – adresování dostupných zdrojů, přijímání požadavků na zdroje, zpracování přijatých a odesílaných dat včetně jejich případné transformace na objekty, volání obslužné operace pro provedení akce spojené s požadavkem, odeslání odpovědi na požadavek s daty navrácenými obslužnou operací;
- Přístup a správa objektů reprezentujících perzistentní data – sada operací pro dotazování a správu dat, která jsou reprezentována výhradně objekty;
- Mapování perzistentních dat na objekty – mapování jednotlivých záznamů z tabulek databáze na objekty.

Architektura části nd-rest-backend se drží uvedeného rozdělení a je dekomponována do tří spolupracujících vrstev. *Vrstva entit* realizuje mapování tabulek databáze na třídy entit pomocí techniky ORM. Přístupem k namapovaným záznamům tabulek a jejich správou se zabývá *vrstva DAO*, jejíž název je odvozen od návrhového vzoru DAO [58], který třídy a rozhraní vrstvy striktně dodržují. *Vrstva komunikačního rozhraní* má na starosti komunikaci pomocí rozhraní založeného na architektuře REST. Za účelem získání nebo modifikace dat jsou z komunikačního rozhraní volány metody tříd z vrstvy DAO. Klíčové parametry, které vedly k výběru REST architektury pro realizaci komunikačního rozhraní, byly především:

- orientace rozhraní na poskytování dat,
- jednoduchý způsob adresování poskytovaných dat,
- možnost zamezení neoprávněného přístupu k datům,
- možnost reprezentace přenášených dat pomocí datově nenáročného formátu JSON, který lze efektivně deserializovat na objekty klientskými skripty ve webovém prohlížeči,
- rozhraní vhodné k budování webových aplikacích založených na technologii AJAX.

Všechny dílčí komponenty jednotlivých vrstev jsou navrženy za pomocí OOP paradigmatu. Návrh pomocí OOP vedl k již zmíněnému použití ORM techniky ve vrstvě entit.

Každá tabulka prostorové databáze je namapována na samostatnou třídu entit. Pro každou třídu entit existuje třída DAO, která předepisuje všechny operace dostupné nad instancemi třídy entit namapovanými z databáze. Třídy DAO vrstvy ve své vlastní režii spravují připojení k databázi. Třídy komunikačního rozhraní jsou také rozděleny tématicky dle třídy entit, jejichž instance poskytují. Uvedený princip návrhu tříd komunikačního rozhraní odpovídá návrhovému vzoru fasáda.

Nemá smysl, aby byla v rámci systému vyvíjena vlastní knihovna realizující ORM techniku nebo knihovna pro tvorbu webové služby poskytující zdroje dle specifikace architektury REST rozhraní, jelikož existují rámce, které se na uvedené činnosti zaměřují. Byly vybrány rámce JPA a JAX-RS z platformy Java EE [20]. Rámec JPA umožňuje mapovat třídy entit na tabulky v relační databázi, realizovat připojení k databázi, spravovat instance namapovaných tříd entit a dotazovat se nad databází. Rámec JAX-RS je určený pro tvorbu webových služeb. Součástí rámce JAX-RS je funkce pro adresování zdrojů, přijímání požadavků, odesílání objektů, transformace vstupních a výstupní dat atd. Další použité součásti platformy Java EE nejsou podstatné pro návrh systému, a proto je jejich význam a způsob použití demonstrován později v kapitole 6.3.

Strukturu části nd-rest-backend lze pozorovat v diagramu tříd na obrázku B.3 na straně 72. Diagram neobsahuje všechny třídy jednotlivých vrstev, snaží se pouze nastínit význam vrstev a jejich vzájemné vazby.

Mezi vrstvou komunikačního rozhraní a vrstvou DAO lze pozorovat vazbu jednotlivých fasádních tříd na DAO rozhraní. Často používaná obecná funkcionalita je zprostředkována abstraktní třídou `AbstractFacade`, která poskytuje například metody pro výstavbu datových struktur tabulek a stromů.

U vrstvy DAO je stěžejní parametrizované rozhraní `Dao`, které obsahuje definici všech běžných funkcí pro práci s persistentními daty (načítání, modifikace, vyhledání atd.). DAO rozhraní pro jednotlivé objekty tříd entit rozšiřují rozhraní `Dao` a přidávají definici specifických metod pro třídu entit. V podbalíčku `impl` se nachází realizace jednotlivých DAO rozhraní. Aby nebylo nutné v každé DAO třídě opět implementovat metody rozhraní `Dao`, je zde třída `JpaDao`, která implementuje všechny metody rozhraní `Dao` pomocí funkcí z rámce JPA. Ostatní implementace DAO rozhraní poté rozšiřují třídu `JpaDao`.

Vrstva entit obsahuje třídy entit realizující rozhraní `IEntityModel`, které je zavazuje k definici číselného identifikátoru. Třídy entit reprezentující pasportizované prvky, u kterých je evidována jejich poloha, navíc realizují rozhraní `ILocated`. Rozhraní předepisuje metody k získání a nastavení vztažené polohy. Navíc je v rozhraní `ILocated` obsažena metoda `getRelated`, která umožňuje získat rekurzivně všechny logicky závislé objekty různých typů k danému objektu. Kvůli zamezení nekonečného zanoření je metodě předávána množina již prozkoumaných objektů. Pomocí této metody lze ve výsledné aplikaci při označení pasportizovaného prvku na mapě vyznačit jeho závislé prvky. Například při volání metody `getRelated` nad optickým vláknem se jako závislé objekty vyhodnotí vlákna, která jsou s daným vláknem svařena a zapojena ve stejném pasivním optickém rozbočovači. Závislé objekty mohou být i jiného typu, jedná se například o koncovky zakončující optické vlákno aj. V tomto případě je celé prohledávání závislostí ukončeno při nalezení obou konektorů zakončujících vlákno nebo jeho závislá vlákna.

nd-client

Část nd-client představuje prezentační vrstvu systému realizovanou pomocí webové aplikace, která poskytuje přístup k datům a službám ostatních částí systému. Webová aplikace

tvoří uživatelské rozhraní systému, jehož dominantním prvkem je interaktivní mapa. O uživatelském rozhraní dále pojednává kapitola 5.2.2.

Samotná webová aplikace je navržena jako množina HTML dokumentů, které neobsahují přímou návaznost na jinou část systému. Dynamičnost rozhraní a řízení uživatelských akcí řeší klientské skripty, které jsou dodávány webovému prohlížeči jako součást HTML dokumentů. Obsahem klientských skriptů je poté funkcionalita nutná ke splnění následujících úkolů:

- komunikace s ostatními částmi systému za účelem získání/modifikace dat,
- definice objektů pro popis struktury dat přijatých z části nd-rest-backend pomocí popisných datových objektů,
- tvorba a správa interaktivní mapy, mapových vrstev a ovládacích prvků mapy pomocí knihovny OpenLayers,
- tvorba tabulek, formulářů, datových pohledů a dalších komponent uživatelského rozhraní automaticky z popisných datových objektů,
- navigace mezi jednotlivými částmi aplikace,
- atd.

Komunikace s nd-rest-backend částí za účelem získání/modifikace neprostorových dat je navržena s použitím technologie AJAX, což napomáhá snižovat datovou náročnost webové aplikace a zvyšovat míru její dynamičnosti. Jelikož načítání mapových vrstev probíhá v knihovně OpenLayers také asynchronně, jsou HTML dokumenty nezávislé na datech, které vizualizují.

Obsah HTML dokumentů je až na internacionalizaci a předání parametrů URL klientským skriptům statický. Strohé požadavky na dynamičnost obsahu HTML dokumentů lze realizovat libovolným jazykem pro dynamické generování HTML dokumentů, například PHP nebo JSP.

Interakce mezi HTML dokumenty, klientskými skripty a dalšími částmi systému probíhá následovně: uživatelská akce spuštěná z rozhraní hlavní stránky webové aplikace vyvolá asynchronní načtení fragmentu HTML dokumentu, který obsahuje klientský skript pro vykonání obsluhy uživatelské akce. Po dokončení načtení fragmentu se obsah fragmentu vloží do k tomu určené části hlavní stránky, což inicializuje spuštění obsaženého klientského skriptu. Pokud klientský skript potřebuje pro svou činnost nějaká neprostorová data, získá je asynchronním požadavkem na část nd-rest-backend s příslušnými parametry. Sémantika získaných dat je určena předdefinovanými popisnými datovými objekty. Po získání dat sestaví klientský skript uživatelské rozhraní (tabulku, formulář apod.) pomocí příslušných knihoven. Modifikace dat pomocí formulářů probíhá podobně. Po odeslání formuláře jsou klientským skriptem zkontrolovány hodnoty jednotlivých polí formuláře. Omezení pro hodnoty je obsaženo v popisných datových objektech. Pokud jsou hodnoty správné, jsou zabaleny do objektu, který je asynchronně zaslán do části nd-rest-backend.

Na obrázku B.4 na straně 73 je zobrazen zjednodušený diagram tříd části nd-client. Diagram neobsahuje všechny třídy klientských skriptů ani HTML dokumenty, snaží se pouze ukázat princip návaznosti komponent části nd-client a rozdělení do balíčků. Klientské skripty jsou organizovány do tří balíčků:

- Balíček „Popis dat“ obsahuje třídu `DataModel`, která se používá pro vytvoření popisných datových modelů. Popisný datový objekt je vytvořen statickou metodou `create`,

které je předáváno URL odkazující na adresu zdroje dat v části nd-rest-backend. Po vytvoření objektu lze přidávat jeho vlastnosti metodou `addProperty`, vazby pomocí `addPivot` metody a seskupovat vlastnosti do skupin vytvořených metodou `addGroup`.

- Balíček „Interaktivní mapa“ je orientovaný na funkcionalitu spojenou s tvorbou a správou interaktivní mapy. Třída `Map` umožňuje vytvořit interaktivní mapu a vložit do ní mapové vrstvy reprezentované objektem třídy `Map_Layer`. Součástí třídy `Map_Layer` je definice stylu vrstvy. Styl je zapouzdřen třídou `Map_Style`. Geoobjekty obsažené ve vektorových vrstvách jsou zapouzdřeny v objektech třídy `Map_Feature`. Ostatní třídy v tomto balíčku reprezentují ovládací prvky mapy.

- Balíček „Klientská aplikace“ spojuje funkcionalitu předchozích balíčků, realizuje navigaci v systému a poskytuje knihovny pro tvorbu uživatelského rozhraní. Klíčovou třídou je metatřída `App`, která realizuje inicializaci celé aplikace, zobrazování stránek klientské aplikace v bočním panelu aplikace, správu informací o autorizovaném uživateli atd. Obsahem třídy je také objekt třídy `App_Map`, který zahrnuje instanci třídy `Map` a instanci třídy `App_Map_Menu` určenou k tvorbě kontextové nabídky mapy. Ostatní třídy obsažené v balíčku reprezentují převážně knihovny pro tvorbu komponent uživatelského rozhraní. Data pro komponenty jsou z části nd-rest-backend načítána dle adres, které jsou asociovány s popisnými datovými objekty.

5.2.2 Uživatelské rozhraní

Uživatelské rozhraní ve formě internacionalizované webové aplikace je obsaženo v části nd-client. Dominantním prvkem je mapa, která je obsluhována ovládacími prvky. Mezi ovládací prvky patří dialog pro výběr zobrazených mapových vrstev, ovládání přiblížení/oddálení mapy a tlačítko pro rozvinutí hlavní nabídky aplikace.

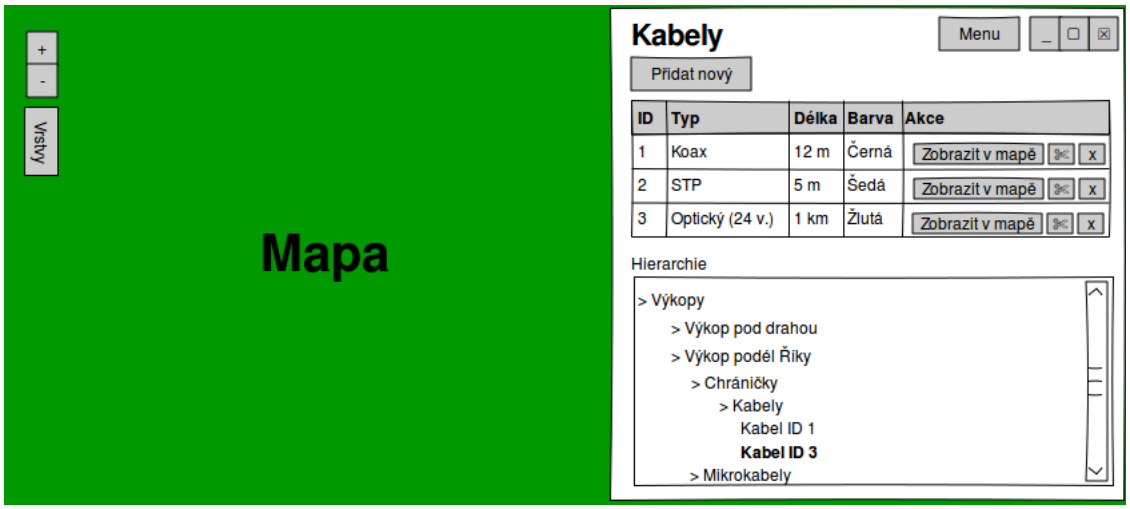
Hlavní nabídka obsahuje obecné položky dostupné uživateli (vyhledání poštovní adresy, odhlášení ze systému, nastavení preferencí, volba jazyka atd.) a položky spojené s pasportizovanými daty. Pro každý typ pasportizovaných dat existuje podnabídka, ve které je dostupná položka pro zobrazení stránky se správou všech prvků tohoto typu a popřípadě další položky poskytující přístup ke specifickým operacím nad prvky tohoto typu.

Obrázek 5.3: Uživatelské rozhraní mapy ve variantě pro desktopový počítač

Většina položek hlavní nabídky vyvolává zobrazení bočního panelu aplikace. Tento panel slouží pro vizualizaci a správu neprostorových dat. Panel je možné maximalizovat přes celou

obrazovku, ale ve výchozím nastavení je zobrazen vedle mapy. Panel obsahuje rozličné komponenty dle zobrazovaného obsahu, nejčastěji tabulky zobrazující data, formuláře pro správu dat, stromové komponenty pro zobrazení hierarchie zanoření prvku atd. Panel je provázán s mapou. Je například možné označit či ukázat na mapě pasportizovaný prvek s definovanou geografickou polohou, který je aktuálně zobrazený v panelu. Provázání je dostupné i opačným směrem pomocí kontextové nabídky mapy.

Kontextová nabídka mapy obsahuje položky dle stavu, jenž je dán skutečností, zda je na mapě vybrán nějaký objekt. Pokud vybrán je, zpřístupňuje kontextová nabídka položky pro jeho modifikaci, přiblížení atd. Nabídka navíc vždy obsahuje položky spojené s přidáváním nových prvků a dodatečnou funkcionalitou mapy. Položka ve většině případech vyvolává zobrazení příslušného obsahu v bočním panelu.



Obrázek 5.4: Uživatelské rozhraní mapy s boční panelem pro zobrazení neprostorových dat ve variantě pro desktopový počítač

Na obrázku 5.3 je zobrazen návrh uživatelského rozhraní aplikace se zavřeným bočním panelem a demonstrací jednotlivých nabídek aplikace. V návrhu bylo pro demonstraci otevřeno několik nabídek, chování nabídek ovšem běžně zamezuje zobrazení více než jedné nabídky současně. Návrh uživatelského rozhraní s otevřeným bočním panelem je dostupný na obrázku 5.3. Boční panel na obrázku obsahuje stránku, která zobrazuje všechny systémem pasportizované kabely včetně zobrazení jejich hierarchie v pasportizované síti.

Představené návrhy jsou určeny pro klasické desktopové PC. Uživatelské rozhraní pro mobilní zařízení je založeno na stejném principu, s výjimkou chování bočního panelu. Na mobilních zařízeních bude panel vždy maximalizován pro reflektování omezené šířky displejů mobilních zařízení. Rozhraní pro mobilní zařízení navíc bude obsahovat rozměrnější ovládací prvky pro snazší ovládání na dotykových displejích. Jelikož rozhraní pro mobilní zařízení není předmětem implementace systému, dále mu není věnována pozornost.

5.2.3 Návrh struktury dat

Návrh struktury dat byl rozdělen, aby byl přehlednější, na část věnující se navržení struktury dat pro pasportizaci a část navržení struktury dat pro správu uživatelských účtů. Navržená struktura dat pro pasportizaci je dostupná ve formě ER diagramu na obrázku B.2 na straně 71. Diagram obsahuje tři druhy entit, které jsou barevně odlišeny.

- Červená entita „pozice“ je jedinou entitou s prostorovými daty (atribut geo). Ostatní

entity, u kterých se udává geografická poloha, se na tuto entitu odkazují. Druhý atribut entity specifikuje typ pozice (bod, křivka). Tento způsob umožňuje, aby dva prvky sítě sdílely stejnou pozici. Neřeší ovšem částečné sdílení polohy, a proto jej řadíme ke špagetovým modelům pro reprezentaci prostorových dat.

- Oranžové entity reprezentují obsah tabulky 4.1 na straně 20. Navíc je zde entita „adresní bod“, která realizuje vztah mezi poštovní adresou a „pozicí“.
- Žluté entity jsou doplňkové entity k oranžovým. Nejedná se o prvky sítě ani o prvky tras, nýbrž o pomocné prvky pro kategorizaci a ukládání dodatečných údajů. Nalezneme zde entity reprezentující typy prvků sítě (např. typ kabelů, typ konektorů) a parametry entity „ostatní“. Atributy entit reprezentujících typy prvků sítě jsou popsány v tabulce 4.1 na straně 20. U prvků typu „ostatní“ lze pomocí parametrů modelovat různé typy síťových prvků (směrovač, přepínač, zesilovač, anténa atd.), což značně snižuje výsledný počet entit.

Všechny entity z návrhu struktury dat pro pasport navíc obsahují primární číselný klíč (ID). V diagramu tyto klíče chybí, jelikož by jejich přidání vedlo k ještě větší velikosti výsledného diagramu a tím i k jeho znepřehlednění.

Návrh struktury dat pro správu uživatelských účtů, uživatelských rolí a persistentní nastavení systému je dostupný ve formě ER diagramu na obrázku B.1 na straně 70.

Struktura takto navržené databáze by se nacházela ve *třetí normální formě*, pokud bychom vynechali tabulku „adresních bodů“. Ta by musela být rozdělena na několik podtabulek k odstranění tranzitivních závislostí mezi atributy (např. mezi městem a státem). Tato tabulka nebyla normalizována, jelikož se v budoucích verzích systému počítá s napojením na IS pro dokumentaci logické struktury počítačové sítě. Tento systém již má adresní body lépe propracované, a proto se při návrhu databáze zvolila dočasná jednodušší podoba tabulky.

5.2.4 Napojení na externí softwarové systémy

Navržený systém je zaměřen na pasportizaci fyzické struktury počítačových sítí. Nelze jej však využít pro pasportizaci logické struktury počítačových sítí ve smyslu evidence IP adres, VLAN sítí, podsítí apod. Již v původním záměru, který vedl k iniciaci vývoje navrženého systému, bylo plánováno napojení systému na systém, který již v současné verzi umožňuje pasportizovat logickou strukturu sítí, zajišťovat účtování za služby, monitorování stavu sítě, správu zákazníků atd. Tímto systémem je otevřený informační systém *FreenetIS* [16].

Implementace napojení navrženého systému na systém FreenetIS je plánována až po dokončení úvodní verze systému a není tedy předmětem této práce. Pro bezproblémovou proveditelnost budoucího napojení je nutné stanovit jeho principy již při prvotním návrhu systému.

Mimo napojení na systém FreenetIS je užitečné zpřístupnit datové zdroje geografického informačního systému jiným GIS nástrojům a využít jejich možnosti například v oblasti analýzy dat.

Napojení na systém FreenetIS

Při napojení navrženého systému a systému FreenetIS je nejdříve nutné vyřešit redundanci prvků jejich datových schémat. Mezi redundantní prvky patří adresní body a „ostatní“ prvky.

- Adresní body, jak už bylo dříve uvedeno, jsou v navrženém systému pouze dočasným řešením. Po napojení na systém FreenetIS budou použity pouze adresní body uložené v systému FreenetIS.
- Tabulka „ostatní“ v navrhovaném systému se po napojení bude dále používat pouze pro pasportizaci prvků, které v systému FreenetIS nelze evidovat.

Po vyřešení redundance schémat je nutné stanovit spojovací bod mezi datovými schématy systému. Systém FreenetIS eviduje fyzické spoje ve formě „linek“, které chápe v případě optických vláken a metalických kabelů jako přenosové médium mezi dvěma porty. Blíže se však nezajímá o strukturu linky, pouze uchovává některé parametry linky (např. maximální přenosovou rychlost). V navrženém schématu databáze lze linku modelovat pomocí dvou libovolně spojených konektorů.[1] Pro realizaci uvedeného spojovacího bodu by do databázového schématu GIS systému byla přidána tabulka „freenetis linka“, obsahem které by byl atribut uchovávající identifikátor linky ze systému FreenetIS. Každá položka této tabulky by obsahovala reference na dva konektory splňující výše uvedené pravidlo.

Propojení systémů pomocí navrženého spojovacího bodu by neprobíhalo na úrovni databází, ale skrze API jednotlivých systémů. Například při zobrazení detailu linky v systému FreenetIS by se systém FreenetIS dotazoval API navrženého systému (což je část nd-rest-backend s přístupem pro externí aplikace), zda neeviduje zobrazovanou linku. Pokud by linka byla evidována, odpověď by obsahovala základní technické údaje o fyzické struktuře linky a odkaz na zobrazení linky v navrženém systému. Poté by uživatel systému FreenetIS měl v zobrazení detailu dané linky zařazeny získané údaje s odkazem pro zobrazení přesnějšího popisu v napojeném systému. Propojení mezi systémy by nebylo pouze jednosměrné, jelikož například vytvoření záznamu v tabulce „freenetis linka“ skrze rozhraní navrhovaného systému by vyžadovalo získání seznamu existujících linek z API systému FreenetIS.

Navržené spojení systémů umožňuje, aby oba systémy koexistovaly a navzájem využívaly svou funkcionalitu a poskytované služby. Navržené propojení lze dále rozvíjet, ale smyslem této podkapitoly bylo pouze ukázat, že je takové propojení vůbec proveditelné.

Napojení na GIS nástroje

Napojení na GIS nástroje je poměrně jednoduché, jelikož k tomuto účelu lze použít WFS a WMS služby, které jsou poskytovány mapovým serverem. Samotné propojení tedy sestává z přidání mapových vrstev mapového serveru do GIS nástroje. Kromě analýzy dat můžeme pomocí takto napojeného GIS nástroje importovat prostorová data do systému. Například pomocí GIS nástroje Quantum GIS (QGIS) lze importovat čáry z CAD výkresu ve formátu DXF do PostGIS databáze. Pasportizace počítačové sítě, která je již dokumentována v CAD výkresu, může být tímto způsobem značně urychlena, jelikož zadávání poloh jednotlivých pasportizovaných prvků patří k časově nejnáročnější etapě pasportizace počítačové sítě pomocí systému. Je nutné podotknout, že ve většině případech je nutné výkresy vhodně upravit, protože výstup z QGIS nástroje je poznamenán nedokonalostmi ve výkresu. Postup úpravy výkresu se značně liší dle jeho zhotovitele a celkové kvality výkresu, a proto nemá smysl jej zde popisovat.

[1] Spojení konektorů lze chápat jako existenci posloupnosti pasivních prvků sítě mezi dvěma konektory. Propojené konektory jsou například konektory zakončující dvě svařená optická vlákna tvořící optický spoj.

Kapitola 6

Implementace

Implementace geografického informačního systému pro pasportizaci počítačových sítí navazuje na kapitolu 5.2, která stanovuje základní aspekty fungování aplikace a dekomponuje ji architektonicky na dílčí podčásti. Tato kapitola si klade za cíl ukázat způsob a prostředky pro realizaci jednotlivých podčástí systému a proces jejich integrace. Pro implementaci celé aplikace byla zvolena platforma Java EE. Důvody a výhody, které vedly k použití této platformy, jsou diskutovány u jednotlivých podčástí systému.

6.1 Prostorová databáze

Jak již bylo uvedeno v kapitole 5.2.1, při implementaci prostorového databázového systému lze použít existující řešení s vlastnostmi definovanými v kapitole 3. Výběr byl proveden z databázových systémů: MySQL, Microsoft SQL Server [3], GeoDB [13], PostGIS [48] a Oracle Spatial. První tři systémy nepodporují transformace mezi souřadnými systémy, a proto nebyly vybrány. Jelikož PostGIS databázový systém je dostupný zdarma i pro komerční využití, byl upřednostněn před Oracle Spatial systémem.

6.1.1 PostGIS

PostGIS je rozšíření objektově-relačního databázového systému PostgreSQL umožňující práci s prostorovými daty. Při práci s PostGIS tedy pracujeme s PostgreSQL databází s přidanými prostorovými funkcemi, datovými typy a prostorovými indexy. PostGIS implementuje geometrický objektový model OGC ukázaný v kapitole 3.1 a také všechny funkce definované nad tímto modelem.[1] Navíc PostGIS obsahuje pokročilé analytické funkce, funkce pro rastrovou mapovou algebru, nástroje pro snadný import a vykreslování dat atd. PostGIS je široce využívaný, což dosvědčuje také podpora pro jeho využití v jiných nástrojích spojených s prostorovými daty [48].

PostGIS funkce a datové typy lze používat v PostgreSQL databázi, která má povoleno rozšíření `postgis`. Sloupec tabulky prostorové databáze obsahující prostorová data lze vytvořit datovými typy `GEOMETRY` a `GEOGRAPHY` nebo dodatečně přidat do tabulky pomocí funkce `AddGeometryColumn`. Pro vytvoření sloupce je nezbytné specifikovat geometrický datový typ sloupce a identifikátor souřadného systému (SRID). Dostupné souřadné systémy spolu s údaji nutnými pro transformace mezi jednotlivými systémy jsou obsaženy v tabulce `spatial_ref_sys`, která je automaticky vytvořena aktivací PostGIS rozšíření. Příklad dota-

[1] PostGIS funkce jsou uvozeny prefixem `ST_`

zování nad daty v takto vytvořené tabulce včetně ukázky transformace je možné pozorovat v kódu 6.1.

```
-- vytvoření sloupce "geom" v tabulce geometry s typem line string
-- se souřadným systémem WGS-84
ALTER TABLE location ADD COLUMN geom GEOMETRY(LINESTRING, 4326);
-- Transformace z WGS-84 do S-JTSK a následný výpočet délky
SELECT ST_Length(ST_Transform(l.geom, 2065)) FROM location l
```

Kód 6.1: Výpočet délky geometrického objektu uloženého v PostGIS databázi se souřadným systémem WGS-84 v metrech

V následujících kapitolách bude ukázáno, jak další části přistupují k datům uloženým v prostorové databázi a jak se nad těmito daty dotazují. Kapitola 6.3.4 popisuje způsob vytvoření schématu databáze dle návrhu struktury dat z kapitoly 5.2.3.

6.2 Mapový server

Podobně jako v případě prostorové databáze není nutné implementovat vlastní řešení, ale využít existující, které splňuje následující podmínky:

- poskytnutí rastrových dat pro čtení skrze WMS služby,
- poskytnutí vektorových dat pro čtení i zápis skrze WFS služby,
- použití PostGIS databáze jako zdroje dat,
- zprostředkování WMS služeb externích mapových serverů včetně uložení zprostředkovaných dat do mezipaměti,
- omezení přístupu k WFS službě dle uživatelských práv,
- otevřený software.

Stanovené podmínky splňují mapové servery MapServer [31] a GeoServer [67]. V kombinaci s PostGIS databází je GeoServer rychlejší [2] a způsob jeho konfigurace je uživatelsky příjemnější. Pro tvořený systém je také výhodou, že GeoServer je Java EE aplikace a lze jej tedy provozovat na stejném aplikačním serveru jako ostatní implementované části systému. MapServer je naopak implementován pomocí CGI rozhraní. Z těchto důvodů byl zvolen GeoServer.

6.2.1 GeoServer

GeoServer je mapový server ve formě webové aplikace, která je organizací OGC označena za referenční implementaci služeb WMS, WFS a WCS. Většina nastavení GeoServeru probíhá skrze webové rozhraní. Jednotlivé poskytované služby jsou shlukovány do pracovních prostorů, které obsahují vrstvy a úložiště. Vrstva předepisuje informace o poskytovaných datech (např. název vrstvy, seznam atributů, souřadný systém, hranice vrstvy, způsob publikování, nastavení vyrovnávací paměti atd.) a odkazuje na úložiště dat. Úložiště je abstrakce pro libovolný zdroj geografických dat. Může se jednat o prostorovou databázi, WMS či WFS službu, různé zdroje rastrových dat apod. Konfigurace úložiště poté zahrnuje nastavení přístupu ke zdroji. Pro PostGIS databázi lze například použít připojení definované na Java

EE aplikačním serveru odkazované pomocí JNDI [20]. Klienti přistupují k vrstvám skrze URL, které se skládá z adresy GeoServeru, URI pracovního prostoru a jména vrstvy. Následná komunikace již probíhá dle publikačního protokolu vrstvy. Mimo základní nastavení umožňuje GeoServer shlukovat vrstvy do skupin pro snazší správu, nastavovat styly vektorových vrstev, spravovat vyrovnávací paměť, spravovat uživatele a jejich skupiny včetně přístupových práv atd.

Z pohledu implementované aplikace obsahuje GeoServer dva pracovní prostory. První pracovní prostor poskytuje rastrové vrstvy reprezentující katastrální mapy ČR skrze WMS služby. Zdrojem konfigurovaným v úložišti těchto vrstev je externí WMS služba poskytovaná veřejně organizací ČÚZK. Nastavení těchto vrstev musí brát zřetel na omezenou rychlost zdroje dat, a proto obsahuje vhodné nastavení vyrovnávací paměti. Druhý pracovní prostor poskytuje skrze WFS služby vektorové vrstvy obsahující pasportizované prvky počítačové sítě, které mají udanou geografickou polohu. Zdrojem dat je tedy PostGIS databáze, jejíž konfigurace je obsažena na Java EE aplikačním serveru a přístup pro GeoServer umožněn skrze JNDI. Oproti přímému přístupu do databáze, který je také možný, je skrze aplikační server sdíleno připojení mezi ostatní části implementované aplikace, což umožňuje snazší rozdělení požadavků na databázový server. V implementované aplikaci chceme kategorizovat tabulku `POZICE` dle prvků, které jsou na ní umístěny. V GeoServeru může být zdrojem dat pro jednu vrstvu pouze jedna databázová tabulka a je tedy nutné kategorizaci provést pomocí databázových pohledů, které musí být z důvodů možnosti modifikace skrze Transactional WFS vybaveny pravidly pro přidávání a modifikaci položek pohledu. Pro možnost přidávání nových záznamů je rovněž nutné vytvořit v databázi tabulku, která bude určovat pro jednotlivé pohledy primární klíče a jejich generátory. Tímto způsobem lze libovolně z jedné tabulky vytvářet v GeoServeru vrstvy s kategorizovanými daty tabulky.

6.3 nd-rest-backend

Jak již bylo uvedeno v kapitole 5.2.1, část nd-rest-backend je třívrstvá serverová aplikace, která spravuje data z PostGIS databáze. Spravovaná data jsou poskytována ve formě objektů skrze rozhraní založené na REST architektuře. Cílem kapitoly je popsat způsob implementace jednotlivých vrstev části nd-rest-backend pomocí Java EE platformy, která byla pro její implementaci zvolena. Vzájemná kooperace vrstev již byla popsána v kapitole 5.2.1, a proto se jí tato kapitola nebude dále věnovat.

6.3.1 Vrstva entit

Vrstva entit obsahuje třídy entit, které jsou namapovány na jednotlivé tabulky v PostGIS databázi technikou ORM pomocí JPA rámce. Jelikož správě instancí tříd entit se věnuje vrstva DAO, použití JPA rámce ve vrstvě entit spočívá především v konfiguraci ORM mapování. Postup mapování je ukázán v následující podčásti kapitoly, která se věnuje popisu tvorby tříd entit. Konec této kapitoly určuje implementaci JPA rámce, která byla vybrána pro použití v implementovaném systému.

Mimo samotné mapování je JPA rámec použit k dotazování nad objekty tříd entit pomocí jazyka JPQL [20]. JPQL je jazyk podobný jazyku SQL, oproti SQL ale umožňuje dotazy aplikovat v nezměněné podobě na rozsáhlou škálu databázových systémů, čímž JPQL zvyšuje přenositelnost celé aplikace. JPQL dotazy jsou v implementovaném systému uchovávány v třídách entit a prováděny ve vrstvě DAO.

Tvorba tříd entit

Třída entit je v kontextu JPA rámce běžná třída jazyka Java opatřená anotací `@Entity`. Jedna či více tabulek databáze může být namapována na jednu třídu entit pomocí anotací `@Table` a `@SecondaryTable`. Jednotlivé sloupce tabulek jsou mapovány na vlastnosti třídy entit. Nastavení mapování sloupců na vlastnosti je provedeno anotacemi aplikovanými na jednotlivé vlastnosti třídy. Vztahy mezi objekty tříd entit jsou vyjádřeny speciálními vlastnostmi objektů, které obsahují referenci na vztažený objekt třídy entit nebo skupinu (kolekci) objektů. Reference mezi objekty může být jednosměrná nebo obousměrná a je explicitně určena dle kardinality vztahu pomocí anotací (`@OneToOne` – 1:1, `@OneToMany` – 1:N, `@ManyToOne` – N:1, `@ManyToMany` – M:N).

Zpravidla používáme JPA rámec ve spolupráci s rámcem Bean Validation [20], který umožňuje na data obsažená v objektech tříd entit aplikovat omezení, pomocí nichž lze zajistit integritu dat. Omezení jsou podobně jako ORM mapování zadávána anotacemi. Kromě existujících předdefinovaných omezení lze definovat i omezení vlastní. Před uložením do databáze jsou tato omezení ověřována, což pomáhá zvyšovat integritu spravovaných dat.

V kódu 6.2 je ukázka částečné definice tříd entit pro databázové tabulky `POZICE` a `VYSTRAZNE_FOLIE` včetně omezení z Bean Validation rámce (`@NotNull`, `@Min` atd.).

```
@Entity @Table(name="POZICE") // třída entit pro tabulku POZICE
public class Location implements IEntityModel {
    @Id private Long id; // primární klíč
    @Enumerated(EnumType.STRING) @NotNull
    private LocationTypeEnum type; // výčtový typ určující typ polohy
    @NotNull @Type(type="org.hibernate.spatial.GeometryType")
    private Geometry geom; // namapovaný sloupec s prostorovými daty
    @OneToMany(mappedBy = "location", cascade = CascadeType.REMOVE)
    private Set<WarningFoil> warningFoils; // výstražné fólie na této pozici
    // další vztahy, vlastnosti, gettery, settery, ...
}
@Entity @Table(name="VYSTRAZNE_FOLIE") // třída entit pro tab. VYSTRAZNE_FOLIE
public class WarningFoil implements IEntityModel, ILocated {
    @Id private Long id; // primární klíč
    @JoinColumn(name = "warning_foil_type_id", nullable = false)
    @ManyToOne(optional = false)
    private WarningFoilType type; // výstražná fólie má povinný typ
    @Min(0) private Integer depth; // hloubka uložení fólie >= 0
    @JoinColumn(name = "location_id", nullable = false)
    @ManyToOne(optional = false) private Location location;
    @ManyToMany private Set<Route> routes; // uložena fólie ve více trasách
    // další vztahy, vlastnosti, gettery, settery, ...
}
```

Kód 6.2: Ukázka definice třídy entit pro databázové tabulky `POZICE` a `VYSTRAZNE_FOLIE`

Hibernate

Rámec JPA je podobně jako většina ostatních komponent platformy Java EE pouze standardizované API. Existuje několik implementací JPA rámce. Při výběru implementace byl kladen důraz především na možnost mapování prostorových datových typů a prostorových operací z PostGIS databáze. Jelikož rámec JPA žádným způsobem nedefinuje spolupráci s prostorovými databázemi, je nutné použít implementaci, která JPA rámec o tyto mož-

nosti rozšiřuje. Jedinou implementací splňující tyto předpoklady je rámec Hibernate [34] s rozšířením **Hibernate Spatial** [36]. S Hibernate Spatial lze prostorové datové typy namapovat pomocí aplikace anotace `@Type` na vlastnost třídy entit, což je patrné z kódu 6.2 na řádku 6. Hibernate Spatial také zprostředkovává prostorové funkce PostGIS databáze do JPQL dotazovacího jazyka. Práce s prostorovými daty je díky této integraci v JPQL podobná práci s neprostorovými daty. S výjimkou těchto rozšíření nejsou v aplikaci využívány další pro Hibernate specifické funkce. Tento přístup usnadní budoucí možnou migraci na jinou implementaci rámce JPA.

6.3.2 Vrstva DAO

Objekty obsažené ve vrstvě DAO musí být schopné řešit souběžný přístup k perzistentním objektům vrstvy entit, jejich načítání z databáze a uložení pozměněných objektů zpět do databáze. Uvedený způsob přístupu si žádá využití transakcí. DAO objekty tedy musí obsahovat podporu pro řízení transakcí. S každým použitím DAO objektů ve vrstvě komunikačního rozhraní není vhodné vytvářet nové objekty, jelikož se ve své podstatě jedná o bezstavové objekty, které je možné sdílet pro více použití a tím šetřit systémové zdroje. Při práci s databází není efektivní k ní inicializovat připojení před započetím každé operace, ale sdílet jej mezi voláním operací a spravovat připojení odděleně. Zmíněné principy DAO objektů lze realizovat pomocí technologie Enterprise Java Bean [20] a rámce JPA.

Enterprise Java Beans (EJB)

V Java EE platformě reprezentuje technologie EJB nástroj pro implementaci aplikační vrstvy. Hlavní snahou EJB je poskytnout generickou implementaci služeb: transakční zpracování, řízení souběžného přístupu, přístup ke zdrojům, distribuovanost, interoperabilita, časované události a mnohé další tak, aby se vývojář mohl soustředit pouze na implementaci aplikační logiky [20]. Samotná implementace aplikační logiky je složena z množiny EJB komponent, které jsou spravovány EJB kontejnerem. EJB komponenta je realizována rozhraním a jeho implementací. Rozhraní EJB komponenty je rozhraní z jazyka Java opatřené anotací `@Remote` pro EJB komponenty, ke kterým lze přistupovat vzdáleně z jiného Java Virtual Machine (JVM), nebo anotací `@Local` pro lokální EJB komponenty. V implementované aplikaci jsou rozhraní EJB komponent z vrstvy DAO označeny jako lokální, jelikož v současném stavu není potřeba je sdílet s jinými aplikacemi.

Implementace EJB komponent jsou založeny na Java Bean [20] třídách, které umožňují pojmenování svých instancí a generický způsob přístupu k vlastnostem instancí. Díky pojmenování lze k instancím přistupovat pomocí JNDI a vkládat závislosti mezi objekty pomocí techniky Dependency injection (DI) [20]. Různé objekty poté mohou přistupovat touto technikou k lokálním či vzdáleným objektům, což zvyšuje interoperabilitu, škálovatelnost a znovupoužitelnost komponent systému.

EJB komponenty dělíme na Session beans a Message driven beans [20]. Pro implementovaný systém jsou podstatné pouze Session beans, které dále rozdělujeme na:

- Stateful Session Beans (stavové) – sdílejí stav mezi voláním svých metod. Pro každého klienta je vytvořen vlastní objekt. Lze je definovat pomocí anotace `@Stateful`.
- Stateless Session Beans (bezstavové) – mezi jednotlivými voláními si oproti stavovým neukládají žádný stav, což umožňuje používat jeden objekt pro obsluhu více klientů nebo naopak více objektů pro obsluhu jednoho klienta, čímž napomáhají zvyšovat škálovatelnost aplikace. Jsou označeny anotací `@Stateless`.

- Singleton Session Beans – jsou založené na návrhovém vzoru jedináček a umožňují udržovat globální sdílený stav aplikace. Pro jejich označení slouží anotace `@Singleton`.

Aplikační logika se vkládá do metod EJB komponent. Všechny metody, které jsou definované v rozhraní EJB komponenty, jsou prováděny v transakcích a je k nim řízen souběžný přístup. Pro definici vlastního prostředí určeného k volání metody EJB komponent lze použít interceptor [20]. V implementované aplikaci se například používá interceptor, který během volání některých metod zachytává a transformuje některé výjimky způsobené voláním dané metody.

Vrstva DAO používá pro implementaci svých rozhraní bezstavové Session beans. Jak bylo naznačeno v kapitole 5.2.1, rodičovské rozhraní `Dao` je implementováno třídou `JpaDao`, která jeho jednotlivé metody realizuje ve spolupráci s rámcem JPA. Pro manipulaci s objekty namapovaných tříd entit definuje JPA rámec správce entit (`EntityManager`). Správce entit je konfigurován pomocí souboru `persistence.xml`, který definuje název, spravované třídy entit, konfiguraci přístupu k použité databázi (v Java EE obvykle přes JNDI), nastavení poskytovatele transakčního zpracování a další dodatečné parametry pro JPA implementaci. Správce entit není v Java EE aplikaci inicializován přímo, ale pomocí techniky DI skrze anotaci `@PersistenceContext`. Příklad vytvoření závislosti lze pozorovat v kódu 6.3. Jednotlivé metody rozhraní `Dao` jsou implementovány v abstraktní třídě `JpaDao` s využitím metod navázaného správce entit. Implementace EJB komponent realizujících metody pro práci s jednotlivými entitami mají díky dědičnosti ze třídy `JpaDao` obecně implementovány metody rozhraní `Dao` a pro implementaci jejich specifických metod je jim k dispozici přístup ke správci entit. Tato architektura vede k redukci výsledného zdrojového kódu a k jeho jednodušší údržbě.

```
1  public interface Dao<T extends IEntityModel> {
2      public Long create(T entity);
3  }
4  public abstract class JpaDao<T extends IEntityModel> implements Dao<T> {
5      @PersistenceContext(unitName = "nd") protected EntityManager em;
6      @Override public Long create(T entity) {
7          Objects.requireNonNull(entity); em.persist(entity);
8          return entity.getId();
9      }
10 }
11 @Local public interface CableDao extends Dao<Cable> {
12     public Collection<Cable> findByMediumType(MediumTypeEnum type);
13 }
14 @Stateless
15 public class CableDaoImpl extends JpaDao<Cable> implements CableDao {
16     @EJB private LocationDao locationDao;
17     public Collection<Cable> findByMediumType(MediumTypeEnum type) {
18         return em.createNamedQuery("Cable.findByMediumType")
19             .setParameter("mediumType", type).getResultList();
20     }
21 }
```

Kód 6.3: Ukázka způsobu implementace EJB komponenty pro správu kabelů včetně podtříd a rozhraní

Ukázka implementace vrstvy DAO na EJB komponentě pro správu entit reprezentujících kabely lze pozorovat v kódu 6.3. Ostatní EJB komponenty jsou implementovány podobným

způsobem. V některých případech je nutné použít metody jiné EJB komponenty. Závislost na používaných komponentách je vytvořena pomocí DI a anotace `@EJB`. Příklad závislosti je dostupný v kódu 6.3 na řádku 15. Vytvoření závislosti na správci entit je možné pozorovat na řádku 5.

6.3.3 Vrstva komunikačního rozhraní

Vrstva komunikačního rozhraní je stejně jako vrstva DAO složena především z EJB komponent, které tvoří jediné ucelené a jednotné rozhraní pro přístup vnějších klientských aplikací do části nd-rest-backend systému. Oproti EJB komponentám vrstvy DAO nemohou přímo přistupovat ke službám rámce JPA. Přístup je umožněn právě skrze EJB komponenty vrstvy DAO. Závislosti na použité DAO EJB komponenty jsou definovány pomocí DI. Komunikační rozhraní nejčastěji komunikuje výměnou objektů tříd entit. Ve složitějších případech, kdy komunikující klient potřebuje zaslat nebo přijmout více objektů různých typů, je zpráva zabalena do obalovacího objektu (wrapper) obsahujícího jednotlivé objekty. Mezi důležité obalovací objekty patří například objekty pro reprezentaci hierarchických stromů dle návaznosti prvků sítě, obalovací objekty reprezentující obsah datových tabulek nebo obalovací objekty reprezentující obsah složitějších formulářů.

Komunikační rozhraní části nd-rest-backend umožňuje klientovi získávat, vytvářet, modifikovat a mazat (CRUD operace) perzistentní objekty namapované z databáze. Rozhraní navíc zprostředkovává dodatečné funkce nad perzistentními objekty, jako je například získání přímo závislých prvků sítě na daném prvku. Pro poskytnutí výše uvedených dat je nutné stanovit jejich adresaci. Jak již bylo zmíněno v kapitole věnované návrhu, byla zvolena architektura rozhraní REST, které pro adresování využívá URL a pro komunikaci protokol HTTP. Pokud chceme nad komunikačním rozhraním budovat klientské aplikace, které mohou být pro jednotlivé datové objekty částečně generické, je nutné adresovat přístup k datovým zdrojům, které poskytují objekty podobným způsobem. V REST rozhraní je adresace možná pomocí kombinace URL a metody HTTP požadavku. Pro adresaci bylo zvoleno následující schéma:

- Všechna URL spojená s objekty jedné třídy entit jsou uvozená názvem dané třídy entit. Tudíž data spojená s třídou entit `Cable` budou dostupná na `<ROOT>/cable/**`. Toto URL bude dále označováno jako kořenové.
- Všechny objekty třídy entit lze získat na kořenovém URL metodou GET. Nepovinně lze také určit pořadí pomocí URL parametrů s názvem sloupce a typem řazení.
- Objekt třídy entit pro přidání lze zaslat na kořenové URL metodou POST. V odpovědi je zaslán identifikátor nově přidaného objektu.
- Objekt třídy entit s modifikovanými vlastnostmi lze zaslat na kořenové URL metodou PUT.
- Objekt třídy entit lze získat na URL zakončeném identifikátorem objektu metodou GET.
- Objekt třídy entit lze smazat na URL zakončeném identifikátorem objektu metodou DELETE.
- Obalovací objekt se seznamem objektů třídy entit, které mohou být seřazeny a filtrovány dle URL parametrů, lze získat na URL zakončeném řetězcem `grid` metodou GET. Tento obalovací objekt slouží jako datový zdroj při tvorbě tabulek.

- Obalovací objekt s jednou úrovní stromu logicky navázaných prvků sítě na rodičovský objekt třídy entit lze získat na URL zakončeném identifikátorem rodičovského objektu a řetězcem `tree` metodou GET. Součástí obalovacího objektu jsou URL definující odkazy s adresami na zdroje dalších úrovní stromu.

- Počet objektů třídy entit lze získat na URL zakončeném řetězcem `count` metodou GET. Nežádoucí objekty mohou být odfiltrovány pomocí URL parametrů.

Další data jsou již libovolně dostupná dle specifičnosti jednotlivých tříd entit. Obvykle se jedná o získání objektů vázaných na daný objekt třídy entit.

Jednotlivé fasádní třídy z vrstvy komunikačního rozhraní poskytují data spojené s třídou entit dle výše popsaného schématu. Zdrojem dat pro tyto třídy jsou EJB komponenty vrstvy DAO. Realizace komunikačního rozhraní je provedena pomocí rámce JAX-RS. Zbytek kapitoly popisuje tvorbu rozhraní pomocí JAX-RS.

Implementace fasádní třídy komunikačního rozhraní pomocí JAX-RS

Při implementaci komunikačního rozhraní postaveného na architektuře REST pomocí rámce JAX-RS se metody běžných tříd jazyka Java mapují jako součásti komunikačního rozhraní. Parametry metody simulují vstupní data a návratová hodnota metody výstupní data. Při tvorbě mapování používáme anotace, které stanovují URL, pod kterým je metoda dostupná, a akceptovanou HTTP metodu. Povolené formáty pro vstupní a výstupní data jsou definované taktéž anotacemi. O vstupně/výstupní transformaci objektů z/do datového formátu se stará technologie Java Architecture for XML Binding (JAXB) [20]. Řízení a správa samotné komunikace probíhá v režii JAX-RS. Při tvorbě komunikačního rozhraní pomocí JAX-RS je tedy z pohledu tvůrce nutné správně nastavit jednotlivé třídy tvořící rozhraní pomocí anotací a implementovat jejich metody. Jelikož není nezbytné se věnovat implementaci samotné komunikace, tělo metod obsahuje ve většině případech pouze aplikační logiku – přímou nebo zprostředkovanou.

V případě implementovaného systému je aplikační logika nejčastěji zprostředkovaná z vrstvy DAO. Jednotlivé fasádní třídy jsou kvůli nutnosti vysoké škálovatelnosti rozhraní implementovány jako bezstavové EJB komponenty. Příklad fasádní třídy s nastavením jejich metod pomocí anotací je dostupný v kódu 6.4. Na řádce 3 je patrná vazba na EJB komponentu z vrstvy DAO. Metoda pro uložení předaného objektu na řádcích 4 až 7 je adresována na kořenovou adresu (adresa definovaná anotací `@Path` na řádku 1) a je ji možné provést metodou POST. Předávaný objekt je očekáván ve formátu JSON. Ostatní metody realizují další části komunikačního rozhraní dle výše uvedeného schématu.

JAX-RS je pouze standardizované API s několika existujícími implementacemi. Jelikož se fasádní třídy striktně řídí standardizovaným API, byla zvolena referenční implementace Jersey [45].

```
@Stateless @Path("cable")
public class CableFacadeREST extends AbstractFacade {
    @EJB private CableDao dao;
    @POST @Consumes({"application/json"}) @Produces({"application/json"})
    public IdWrapper create(CableAddFormWrapper entity) {
        return cidw(dao.createFromForm(entity));
    }
    @PUT @Consumes({"application/json"})
    public void edit(Cable entity) {
        dao.update(entity);
    }
    @DELETE @Path("{id}") public void remove(@PathParam("id") Long id) {
        dao.remove(id);
    }
    @GET @Path("{id}") @Produces({"application/json"})
    public Cable find(@PathParam("id") Long id) {
        return dao.find(id);
    }
}
```

Kód 6.4: Ukázka způsobu implementace fasádní třídy komunikačního rozhraní pomocí JAX-RS rámce

6.3.4 Inicializace a povýšení schématu databáze

Při nasazení systému je nutné řešit způsob prvotní inicializace schématu databáze a jeho dodatečné povýšení plynoucí z možné změny struktury dat po vydání nové verze systému. Inicializace i povýšení schématu lze realizovat například pomocí skriptů obsahujících potřebné definice v jazyce SQL. Ve snaze co nejvíce usnadnit přechod na novou verzi systému je vhodné vybudovat mechanismus, který při nasazení na aplikační server provede SQL skripty bez vnějšího zásahu.

Mechanismus pro produkční prostředí

V implementovaném systému se o inicializaci a povýšení schématu databáze stará část nd-rest-backend. Mechanismus je implementován pomocí bezstavové EJB komponenty opatřené anotacemi `@Singleton` a `@Startup`, které zapříčiní, že metoda komponenty opatřená anotací `@PostConstruct` se provede během zavádění části nd-rest-backend na aplikační server. Pro možnost povýšení je dále nutné stanovit pevný formát a místo uložení pro aktuální verzi systému a verzi schématu databáze. Ideální formát verze je složený z čísel a oddělovačů, jelikož verze lze v takovém formátu vzájemně seřazovat a tím umožnit postupné povýšení o několik verzí. Verzi systému lze ukládat například v konstantě uvnitř zdrojového kódu systému. Verzi schématu databáze je vhodné ukládat přímo v databázi. V implementovaném systému pro tyto účely slouží záznam s názvem `schema_version` v tabulce `NASTAVENI_SYSTEMU`. Součástí části nd-rest-backend jsou SQL skripty pro povýšení schématu pojmenované dle verze, pro kterou je povýšení vyžadováno. Navíc existuje také skript pro prvotní inicializaci schématu databáze. Během zavádění části nd-rest-backend se poté zkontroluje, zda odpovídá hodnota `schema_version` aktuální verzi systému, pokud ano, zavádění pokračuje dále. Pokud naopak není tabulka `NASTAVENI_SYSTEMU` v databázi vůbec nalezena, je spuštěn skript pro prvotní inicializaci. V případě, že je verze schématu načtena,

ale je vyšší než verze aplikace, zavádění je ukončeno s chybou, jelikož byla verze systému snížena, což není povoleno. V posledním případě, kdy je načtená verze schématu nižší, jsou vyhledány všechny dostupné skripty pro povýšení mezi oběma verzemi a následně jsou v seřazeném pořadí provedeny. Důležité je také zmínit, že každé povýšení a také inicializace aktualizuje hodnotu verze schématu databáze.

Mechanismus pro testovací prostředí

Dalším možným automatizovaným přístupem k implementaci mechanismu je použití rámce Hibernate, který umožňuje vytvářet a povyšovat schéma automaticky dle definice mapování v třídách entit. Takové řešení je nevhodné pro produkční nasazení systému, jelikož není zajištěno, že složitější povýšení bude provedeno bez chyb a neovlivní v databázi uložená data. Naopak toto řešení je výhodné během testovaní systému s častými změnami schématu.

6.4 nd-client

Nejdůležitější částí implementace nd-client jsou klientské skripty, které realizují interaktivní mapu, napojení mapy na aplikaci v bočním panelu, která bude dále označována jako klientská aplikace pro správu neprostorových dat, samotnou klientskou aplikaci a komunikaci s ostatními částmi systému. Tyto činnosti jsou také svázané s tvorbou uživatelského rozhraní. Jelikož se jedná o webovou aplikaci, uživatelské rozhraní je vytvořeno kombinací HTML dokumentů s kaskádovými styly v podobě CSS souborů a klientských skriptů, které jsou vytvořeny pomocí skriptovacího jazyka JavaScript. Následující podpkapitoly se postupně zabývají způsobem generování HTML dokumentů, komunikací s částí nd-rest-backend, realizací interaktivní mapy, představením komponentů pro výstavbu uživatelského rozhraní klientské aplikace a internacionalizací uživatelského rozhraní.

6.4.1 Dynamické generování HTML dokumentů

Obsah HTML dokumentů se až na internacionalizaci a předání parametrů URL klientským skriptům nemění. Generování dokumentů se provádí pomocí **Java Server Pages (JSP)** stránek [20], které jsou součástí platformy Java EE. Technologie JSP byla vybrána především proto, aby bylo možné provozovat nd-rest-backend a nd-client na stejném aplikačním serveru. Bez větších problémů a změn by bylo možné použít pro generování například jazyk PHP.

```
1  <%@page contentType="text/html" pageEncoding="UTF-8"%>
2  <%@taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
3  <fmt:setBundle basename="i18n.base" />
4  <script type="text/javascript">
5      $(document).ready(function () {
6          $("#edit").dataEdit(dm.Cable, "<%= request.getParameter("id") %>");
7      });
8  </script>
9  <h1><fmt:message key="dm.Cable.editTitle" /></h1>
10 <div id="edit"></div>
```

Kód 6.5: Ukázka JSP stránky realizující tvorbu části HTML dokumentu pro editaci kabelu s předaným identifikátorem

Příklad jednoduché JSP stránky pro generování fragmentu HTML dokumentu, realizujícího formulář pro editaci kabelu, je dostupný v kódu 6.5. Identifikátor editovaného kabelu je předán skrze parametr URL `id`. Získání parametru URL je možné pozorovat na řádku 6. Nadpis na řádku 9 je internacionalizován. Detaily ohledně internacionalizace JSP stránek lze nalézt v kapitole 6.4.5. Většina JSP stránek s výjimkou přihlašovací obrazovky a hlavní stránky generuje pouze fragmenty HTML dokumentů, které jsou dynamicky vkládány do hlavní stránky.

6.4.2 Komunikace s nd-rest-backend

Jak již bylo nastíněno v kapitole věnované návrhu, komunikace s ostatními částmi systému a tedy i s částí nd-rest-backend probíhá asynchronně pomocí technologie AJAX. Pro zapouzdření funkcí spojených s vytvářením AJAX požadavků na server, získávání odpovědí od serveru a dalších operací s tím spojených byla s ohledem na komunikační rozhraní části nd-rest-backend vytvořena knihovna `RestClient`. Knihovna umožňuje jednoduše odesílat požadavky na server různými HTTP metodami, konfigurovat zasílané HTTP hlavičky, přikládat k požadavkům data, přijímat odpovědi, zpracovávat data obsažená v odpovědích a řešit chyby vzniklé během komunikace. Všechny komponenty, které komunikují s částí nd-rest-backend používají služby knihovny `RestClient`.

V kódu 6.6 je možné pozorovat způsob práce s knihovnou. Kód ukazuje manipulaci s objekty třídy entit reprezentující kabely. Na řádku 2 získá HTTP požadavkem s metodou GET všechny objekty. Adresa (URL) je získána z popisného datového objektu pro kabely. Následně jsou na řádcích 3 až 5 modifikovány získané objekty a postupně uloženy HTTP požadavkem s metodou PUT na adresu rozšířenou o identifikátor jednotlivého objektu.

```
1 try {
2     var cables = RestClient.create(dm.Cable.url).get();
3     for (var i = 0; i < cables.length; i++) {
4         cable.comment += " - upraveno";
5         RestClient.create(dm.Cable.url+"/"+cable.id).sedData(cable).put();
6     }
7 } catch (exception) { /* řešení vzniklé chyby */ }
```

Kód 6.6: Příklad práce s `RestClient` knihovnou, který ukazuje, jak část nd-client komunikuje s částí nd-rest-backend

Tvorba popisných datových objektů

Objekty získané a zasílané pomocí `RestClient` knihovny reprezentují objekty tříd entit a obalovací objekty z části nd-rest-backend, které postrádají informace o typech jejich vlastností, typech objektů tříd entit, se kterými jsou ve vztahu atd. Uvedené informace jsou však nezbytné pro efektivní práci s těmito objekty, a proto jsou k nim vytvářeny v části nd-client popisné datové objekty. Do jisté míry se jedná o duplikaci tříd entit a tříd obalovacích objektů z vrstev části nd-rest-backend. Navíc popisné datové objekty obsahují konfiguraci pro vytváření komponent uživatelského prostředí. Cílem je, aby pro vytvoření komponenty uživatelského rozhraní postačovalo předat příslušné knihovně pro tvorbu komponenty pouze popisný datový objekt a samotná data nebo reference (adresy) k jejich získání.

Popisné datové objekty lze, jak již bylo uvedeno v kapitole 5.2.1, vytvářet pomocí třídy `DataModel`. K jednotlivým vztahům a vlastnostem popisného datového objektu lze přiřazovat formátovací pravidla určená k převodu hodnoty vlastnosti na řetězec, typ komponenty

pro modifikaci hodnoty vlastnosti ve formulářích, omezení kladená na hodnoty zadané ve formulářích, způsob zobrazení v datové tabulce apod. Komponenty uživatelského rozhraní se tvoří pomocí uvedených dodatečných informací. Například formulář pro editaci dat svázaných s předaným popisným datovým objektem obsahuje pole pro každou vlastnost a vazbu definovanou popisným objektem.

Pro všechny třídy entit části z nd-rest-backend jsou popisné datové objekty vytvořeny během načtení hlavní stránky tak, aby byly dostupné v části nd-client globálně. Globální přístup je umožněn skrze globální proměnnou `dm`. Popisné datové objekty pro obalovací objekty jsou vytvářeny až v případě jejich potřeby.

Kód 6.7 demonstruje vytvoření popisného datového modelu k třídě entit reprezentující kabel. Na řádku 2 je vytvořena instance třídy `DataModel` s kořenovou adresou datových zdrojů spojených s objekty třídy entit reprezentující kabely v části nd-rest-backend. Na řádcích 3 až 9 jsou pomocí metody `addProperty` vytvářeny vlastnosti popisného datového objektu. První parametr metody určuje název vlastnosti, druhý parametr určuje jeho datový typ a třetí parametr obsahuje internacionalizovaný štítek, který je zobrazován například v hlavičkách tabulek, u polí formuláře apod. Čtvrtý nepovinný parametr metody obsahuje objekt, ve kterém je specifikováno dodatečné nastavení vlastnosti. Například na řádku 5 tento parametr specifikuje adresu pro získání seznamu přípustných hodnot určených pro prvek formuláře. Na řádku 5 až 6 je možné si všimnout vlastnosti, která vytváří 1:N vztah mezi objekty reprezentující kabely a jejich typy. Řádek 11 obsahuje ukázku definice M:N vztahu.

```
1  // vytvoření modelu pro kabel s jeho vlastnostmi a 1:N vztahem k typu kabelu
2  dm.Cable = DataModel.create(dm.URL + "/cable", "Cable")
3      .addProperty("id")
4      .addProperty("location", "location_line", _("dm.prop.location"))
5      .addProperty("type", "enum", _("dm.prop.type"),
6                   {otype: "CableType", selectDataUrl: dm.URL + "/cable_type"})
7      .addProperty("color", "color", _("dm.prop.color"))
8      .addProperty("length", "integer", _("dm.prop.length"))
9      .addProperty("comment", "lob", _("dm.prop.comment"));
10 // vytvoření vlastnosti s informací o M:N vztahu kabelu s trasou
11 dm.Cable.addPivot("routes", dm.Route, {title: _("dm.prop.locatedInRoutes")});
```

Kód 6.7: Příklad vytvoření popisného datového modelu pro popis objektů třídy entit reprezentující kabely

6.4.3 Interaktivní mapa – OpenLayers

Interaktivní mapa – nejdůležitější výstup implementovaného systému – je realizována pomocí knihovny OpenLayers. Díky existenci kolekce předdefinovaných ovládacích prvků a vestavěné podpoře pro WMS a WFS služby se jedná o ideální prostředek pro implementaci webově orientovaných GIS systémů. Velkou výhodou je také přehledný volně dostupný zdrojový kód, který není obtížné přizpůsobit potřebám implementovaného systému. V kódu 6.8 je demonstrováno vytvoření jednoduché mapy. Význam jednotlivých částí kódu bude objasněn v průběhu této kapitoly.

Pro ovládání mapy byly použity vestavěné ovládací prvky. Jednalo se například o ovládací prvek určený pro výběr zobrazených mapových vrstev, prvek pro nastavení přiblížení mapy, nástroj pro kreslení a modifikaci vektorových objektů, nástroj pro zobrazení aktuální pozice uživatele využívající HTML5 [35] geolokačního rozhraní, prvek pro zobrazení

souřadnice aktuální polohy, měřítko mapy aj. Kolekci ovládacích prvků lze navíc rozšířit o nové prvky. V kódu 6.8 na řádku 6 je ukázáno vložení ovládacího prvku pro přepínání mezi mapovými vrstvami do mapy vytvořené OpenLayers knihovnou.

```
1  var map = new OpenLayers.Map({
2      div: "map", // ID DOM elementu nad kterým je mapa vytvořena
3      projection: new OpenLayers.Projection("EPSG:900913"),
4      displayProjection: new OpenLayers.Projection("EPSG:5513")
5  });
6  map.addControls([new OpenLayers.Control.LayerSwitcher()]);
7  map.addLayers(new OpenLayers.Layer.Bing({
8      name: "Bing Hybrid",
9      key:  "xxxx",
10     type: "AerialWithLabels",
11     isBaseLayer: true
12 }));
```

Kód 6.8: Příklad tvorby interaktivní mapy pomocí knihovny OpenLayers

OpenLayers ve spolupráci s knihovnou *proj4js* umožňuje pracovat s mapovými vrstvami v různých souřadných systémech a zobrazovat je v různých projekcích. To je velmi užitečné, jelikož jak již bylo zmíněno, výsledná aplikace nutně potřebuje pracovat s mapovými podklady v různých formátech. V kódu 6.8 na řádcích 1 až 5 je ukázka vytvoření mapy s explicitně definovanými mapovými projekcemi.

Možnost přepínat se mezi více mapovými vrstvami umožnila, aby uživateli byly zpřístupněny podkladové rastrové mapové vrstvy z více zdrojů. Primárně se používají detailní volně dostupné OpenStreetMaps, ale uživatel si může vybrat také z různých variant Bing Map. Další podkladové vrstvy lze přidat, ale díky detailnosti OpenStreetMaps map to nebylo prozatím nutné. Mimo běžné podkladové mapy lze volitelně překrýt podkladovou mapu vrstvou s katastrálními mapami získanými z lokálního mapového serveru skrze WMS službu. Způsob konfigurace mapových vrstev je ukázán v kódu 6.8 na řádcích 7 až 12, který obsahuje definici podkladové mapové vrstvy z Bing Map.

Klíčovým prvkem mapy jsou vektorové vrstvy s prvky pasportizované počítačové sítě načítané z WFS služby na lokálním mapovém serveru. Vykreslování jednotlivých poloh je podřízeno typu převládajícího prvku na dané pozici, oddálením mapy a zvoleném módu mapy. Pomocí dynamických stylů z knihovny OpenLayers je docíleno odlišného vzhledu různých prvků, a to nejen odlišením barev, tloušťky čar, typů čár, ale také použitím zástupných symbolů (ikon) pro prvky s bodovou polohou. Místem, kde muselo být chování OpenLayers do značné míry upraveno, je způsob výběru, vytváření a editace objektů vektorových vrstev. Ve výchozím stavu OpenLayers umožňuje tyto metody provádět hromadně a změny v objektech ukládat po dávkách. Pro transakčně orientovaný IS, kde je modifikace či vytvoření svázáno s dalšími neprostorovými daty, je takový postup nevhodný. Bylo proto nutné, aby operace jako přidávání (zakreslování) či editace byla prováděna nad jedním vektorovým objektem a po ukončení operace mohl být objekt samostatně uložen. Totéž platí o výběru objektu, jelikož například při přidání nového síťového prvku do existující polohy musíme uživateli povolit zvolit pouze jednu polohu s typem, který odpovídá typu polohy daného prvku. Dalším rozšířením funkcionality výběru je také nutnost při označení prvku vyznačit na něho navazující prvky. Informace o tom, které prvky jsou návazné, se získá z části nd-rest-backend metodou třídy entit `getRelated` popsané v kapitole 5.2.1. Z důvodů nutnosti specifické funkcionality jsou některé ovládací prvky a také konfigurace mapy, stylů a dalších komponent mapy zapouzdřeny do vlastních tříd a objektů. Na některých

místech muselo dojít k modifikaci samotné OpenLayers knihovny.

Návaznost mapy na klientskou aplikaci je umožněna přidanou funkcionalitou do jednotlivých zapouzdřených tříd a objektů ve formě, která vyhovuje aplikaci. Uživatelské akce spojené s klientskou aplikací jsou ve většině případech realizovány pomocí kontextové nabídky mapy. Položky nabídky se přizpůsobují aktuálnímu vybranému prvku a zprostředkovávají operace jak pro přiblížení, modifikaci, mazání, vložení nových prvků sítě, způsob vyznačení vybraného vektorového objektu mapy, tak i pro globální práci s mapou. Mezi globální akce patří například získání informace z katastru nemovitostí, vložení nového prvku sítě s novou polohou apod.

Mimo použití vestavěných, i když někdy modifikovaných, ovládacích prvků mapy je vytvořen jeden vlastní ovládací prvek. Jedná se o výběr módu mapy, který ovlivňuje styl a způsob vykreslení prvků na mapě. Existují čtyři módy:

- Mód tras – zobrazuje pouze polohy, na kterých se nachází prvky sítě spojené s jejich trasami, tedy samotné trasy, kabelové komory, chráničky, výstražné fólie a sloupy.
- Mód kabelů – vizualizuje polohy s kabely, optickými buffery, kabelovými komorami, sloupy, koncovkami metalických kabelů apod.
- Mód optických vláken – zobrazuje všechny polohy s optickými vlákny, rozbočovači, rozvaděči, optickými konektory, adaptéry atd.
- Kombinovaný mód – vizualizuje všechny polohy a na každou aplikuje styl dle nevýznamnějšího prvku, například poloha, ve které je vložena trasa, kabel i optická vlákna, bude stylizován jako trasa.

Styly pro vykreslení prvků se přizpůsobují zvolenému módu a umožňují uživateli se lépe orientovat v mapě. Výchozím módem je kombinovaný mód, jelikož vždy obsahuje všechny dostupné prvky v dané úrovni zvětšení mapy, což umožňuje komplexní prvotní náhled na data systému.

6.4.4 Klientská aplikace

Klientská aplikace je sestavována z komponent. Základní komponenty jsou tabulky pro zobrazování objektů stejného typu, pohledy pro zobrazení konkrétního objektu, stromové struktury pro zobrazení závislostí mezi objekty, formuláře pro modifikaci objektů a hlavní nabídka. Jednotlivé komponenty tvoří části uživatelského rozhraní klientské aplikace z informací v datových modelech, představených v kapitole 6.4.2, a komunikují s částí ndrest-backend pro získání a ukládání dat. Mezi komponenty patří také tlačítka umožňující přechod mezi stránkami aplikace. Aktuální stránka je zaznamenávána v kotvě URL, jelikož kotevní část URL lze měnit pomocí klientských skriptů bez nutnosti opětovného načtení WWW stránky. Zaznamenání aktuální pozice v aplikaci do kotvy umožňuje, aby uživatel mohl používat funkce webového prohlížeče pro procházení historie a obnovení zobrazené stránky.

Při implementaci všech komponent byla použita knihovna jQuery [15], která umožňuje vytvářet multiplatformní klientské skripty. Výhodou knihovny jQuery je existence široké škály volně dostupných zásuvných modulů pro řešení rutinních problémů při vývoji webových aplikací. Některé dostupné zásuvné moduly byly použity jako samotné komponenty klientské aplikace, jiné pro implementaci částí jednotlivých komponent. U komponent, pro

které neexistoval vhodný modul, byly zhotoveny vlastní jQuery moduly. Spojením komponent vzniká ucelené uživatelské rozhraní s nízkou datovou náročností, jelikož závislosti nutné pro tvorbu komponent jsou načteny pouze jednou a při dalších klientských požadavcích jsou získána pouze data a fragment HTML dokumentu, který po vložení do hlavního dokumentu inicializuje výstavbu komponent. Příklad takového fragmentu je patrný z kódu 6.5 na straně 44. Zbytek této kapitoly obsahuje krátký popis jQuery knihovny, výčet použitých existujících jQuery modulů a popis vytvořených jQuery modulů.

jQuery

jQuery je v současnosti nejpoužívanější knihovna [63] pro tvorbu dynamických webových aplikací v jazyce JavaScript. Klíčové přednosti a funkce, které vedly k širokému rozšíření této knihovny, jsou především selektory podobné selektorům v CSS3 [38] pro výběr, procházení a modifikaci elementů HTML dokumentu, dále zjednodušená správa událostí, zjednodušená manipulace s CSS, zjednodušená práce s technologií AJAX, možnost tvorby animací, široká podpora mezi webovými prohlížeči, volná dostupnost, rozsáhlá kolekce zásuvných modulů atd.

Použité existující zásuvné moduly jQuery

Následující zásuvné moduly jsou již existující řešení, která byla použita pro usnadnění implementace klientské aplikace. Pokud bylo nutné modul nějakým způsobem upravit, je popis úpravy explicitně uveden.

jQuery chosen je modul pro tvorbu políček formulářů s možností výběru jedné či více předvyplněných položek. Oproti běžnému HTML `select` prvku formuláře je s ním práce uživatelsky intuitivnější, například při vyhledávání položek.

jQuery colorpicker umožňuje uživateli vybrat barvu z nabídnuté palety.

jQuery contextMenu je modul pro tvorbu kontextových nabídek. Podstatnou nevýhodou modulu je nemožnost přidávat nové položky do nabídky po její inicializaci. Tato funkcionalita byla nicméně nezbytná, a proto bylo nutné modul rozšířit.

jQuery dynatree slouží pro tvorbu dynamických stromových struktur, jejichž uzly mohou být načítány asynchronně. Asynchronně načtený uzel může obsahovat statické poduzly nebo adresu, na které lze asynchronně získat poduzly. Modul byl rozšířen o integraci s jQuery contextMenu modulem, díky které lze pro uzly budovat kontextové nabídky.

jQuery jqGrid je modul pro tvorbu tabulek s možností asynchronního načítání dat, stránkování, filtrování obsažených záznamů, řazení sloupců atd.

jQuery UI představuje modul, který je spíše knihovnou obsahující kolekci ovládacích prvků (widget), efektů, animací apod. Mezi ovládací prvky zahrnuté v této knihovně patří například dialogy, tlačítka a záložky. Jednotlivé komponenty modulu UI jsou použity v dalších modulech klientské aplikace.

jQuery validate modul umožňuje deklarativním způsobem definovat omezení aplikovaná pro kontrolu uživatelských vstupů ve formulářích. Kromě předdefinovaných pravidel lze definovat i pravidla vlastní.

Zásuvné moduly jQuery vytvořené pro potřeby klientské aplikace

Všechny níže uvedené moduly spolupracují s popisnými datovými objekty. Z informací obsažených v popisných objektech vytváří obsah jedné stránky klientské aplikace (výjimkou je jQuery dataGrid modul).

jQuery dataAdd je modul vytvářející stránku s formulářem pro přidání objektu třídy entit. Stránka vygenerovaná modulem obsahuje nadpis, formulář pro zadaní dat a tlačítko pro přechod na zobrazení všech objektů tříd entit daného datového modelu. Formulář je tvořen kombinací HTML elementů, jQuery chosen komponentů, jQuery colorpicker komponentů a komponentou pro zadání polohy. Validaci hodnot obstarává jQuery validate modul. U prvků sítí s evidencí polohy se používá speciální komponenta, která uživateli umožňuje přiřadit vytvářenému prvku existující polohu nebo inicializovat vytvoření nové polohy. Přiřazení probíhá výběrem prvku na mapě. U jQuery chosen komponent formuláře se jejich hodnoty načítají z části nd-rest-backend během vytváření formuláře. Po úspěšném odeslání dat z formuláře do části nd-rest-backend se uživateli zobrazí odkaz pro zobrazení nově přidaného prvku. Jelikož ne vždy lze některé činnosti spojené s přidáváním nových prvků plně automatizovat, je možné pomocí konfigurace zadávané při inicializaci modulu nastavit specifické chování pro daný formulář. Nastavení umožňuje například vložit funkci pro vlastní validaci dat, funkci pro úpravu dat před jejich odesláním atd.

jQuery dataEdit je modul pro tvorbu editačních formulářů, který je velmi podobný modulu dataAdd. Oproti modulu dataAdd je přidána podpora pro načtení dat editovaného objektu během inicializace formuláře a tlačítko pro zobrazení stránky s detailem editovaného objektu. Editovaný objekt je zadán modulu pomocí jeho identifikátoru.

jQuery dataView je modul pro tvorbu zobrazení stránky s detailem objektu. Stránka s detailem obsahuje nadpis, tabulku obsahující vlastnosti objektu, tlačítka pro editaci a mazání objektu a záložky se závislými prvky. Obvykle se na stránce s detailem zařízení vyskytují dvě záložky: první se stromovou strukturou návazných prvků a druhý s tabulkou přímo návazných prvků, které jsou například v daném prvku přímo obsaženy. Pro tvorbu stromu je použit modul jQuery dynatree s kořenem v zobrazovaném objektu. Jednotlivé tabulky jsou následně vytvářeny pomocí dataGrid modulu s explicitním určením adresy závislých dat v části nd-rest-backend. Modul podporuje přidávání záložek s dalším rozličným obsahem, který je specifický pro daný objekt.

jQuery dataGrid je modul pro tvorbu jedné datové tabulky, tlačítek k provázání tabulky s aplikací a kontextové nabídky k položkám tabulky. Pro tvorbu samotné tabulky se používá modul jQuery jqGrid a pro kontextovou nabídku modul jQuery contextMenu. Úkolem modulu je především konfigurace a inicializace jqGrid tabulky. Výsledná tabulka může být použita jako komponenta stránky například u zobrazení detailu objektu nebo jako samostatná stránka zobrazující všechny objekty třídy entit daného popisného datového modelu. Dle použití se liší také dostupná tlačítka. V případě použití na celou stránku je obvykle dostupné tlačítko pro přidání nové položky. U použití jako komponenty je k tomuto tlačítku možné připojit informace o objektu, na jehož stránce je příslušná tabulka zobrazena. Připojené informace slouží během přidání pro předvyplnění závislého prvku v poli příslušného formuláře.

6.4.5 Internacionalizace

Při návrhu a implementaci mechanismu pro internacionalizaci uživatelského rozhraní části nd-client byly kladeny následující požadavky:

- prostředky pro internacionalizaci musí být dostupné během generování HTML dokumentů a jejich fragmentů JSP stránkami a během provádění klientských skriptů;
- musí být umožněno internacionalizovat odpovědi z části nd-rest-backend (např. chybová hlášení);
- překladové řetězce musí být sdíleny mezi všemi případy užití internacionalizace.

Pro internacionalizaci v rámci JSP stránek byla použita podčást standardní knihovny značek JSTL [20], označovaná jako *fmt* [17]. Kromě funkcionality pro formátování obsahuje fmt rovněž možnost internacionalizace pomocí řetězců ze souborů typu properties. Soubor typu properties obsahuje na svých jednotlivých řádcích klíč a jeho hodnotu. V případě překladových řetězců je poté klíč použit pro identifikaci internacionalizovaného řetězce a hodnota je samotný překladový řetězec. Pro každý podporovaný jazyk existuje samostatný soubor typu properties se stejnými klíči. Důležitou funkcí je možnost předávání parametrů do internacionalizovaných řetězců. Ukázka překladu pomocí značek z jmenného prostoru `fmt` je možné pozorovat v kódu 6.5 na straně 44.

Řetězce, které nemohou být internacionalizovány v době generování HTML dokumentů JSP stránkami, jsou internacionalizovány klientskými skripty pomocí funkce `_`, které je předán klíč a nepovinně také parametry pro vložení do internacionalizovaného řetězce. Použití funkce je patrné z kódu 6.7 na straně 46.

Jelikož překladové soubory jsou umístěny v části nd-client, nemá k nim část nd-rest-backend přístup. Některé odpovědi (například chybová hlášení) je ovšem nutné internacionalizovat. Řešením je do řetězců obsažených v odpovědích z části nd-rest-backend umísťovat značky odkazující se na klíče překladových řetězců z části nd-client. Po přijetí dat z části nd-rest-backend mohou být řetězce v části nd-client intenacionalizovány. Byly zvoleny značky ve formátu `#{<klíč>}`. Řetězec může obsahovat libovolný počet těchto značek, ale bez možnosti jejich zanoření. Nahrazení značek za překladové řetězce je realizováno pomocí výše uvedené funkce `_`.

Hlavní stránka nd-client obsahuje možnost přepínat se mezi jazyky pomocí URL parametru `lang`. Výchozí jazyk je určen dle HTTP hlavičky `language` zaslané webovým prohlížečem. V současné době jsou dostupné lokalizace pro český a anglický jazyk.

6.5 Autentizace a autorizace uživatelů

Implementovaný systém musí povolovat přístup k jednotlivým svým částem a službám pouze pro autentizované a autorizované uživatele. Jelikož vstupním bodem do systému pro uživatele je jeho část nd-client, nachází se zde také formulář pro autentizaci uživatele. Autentizační údaje uživatele jsou ověřovány v části nd-rest-backend, ve které jsou uloženy uživatelské účty. Po úspěšné autentizaci uživatele mu musí být k dispozici služby části nd-rest-backend a mapového serveru, ke kterým je autorizován po dobu platnosti autentizace. Přístup k databázi z mapového serveru i z části nd-rest-backend je globálně řízen aplikačním serverem pomocí JNDI.

Pro implementaci autorizace a autentizace části nd-rest-backend je použita knihovna

Spring Security [65]. Jedná se o pokročilou knihovnu, kterou lze použít jen s minimem zásahů do samotné aplikace. Většina úkonů nutných pro její integraci je spojena s tvorbou konfiguračního XML dokumentu, kde se definuje služba pro správu autentizace, způsob hašování hesel, služba poskytující údaje o uživatelích, seznam pravidel definujících omezení přístupu a role nutné pro autorizaci uživatelů k nim atd. Kromě vlastní implementace služby poskytující údaje o uživatelích a úpravě entity reprezentující uživatele nebylo nutné zasahovat do zdrojových kódů. Implementace služby poskytující uživatelské údaje je provedena ve vrstvě DAO pomocí EJB komponenty rozšiřující rozhraní `UserDetailService`, které deklaruje metodu pro získání objektu třídy entit reprezentující uživatele zadáním jeho uživatelského jména. Třída entit reprezentující uživatele musí implementovat rozhraní `UserDetail`, které deklaruje metody pro získaní hesla, uživatelského jména, seznamu přiřazených systémových rolí uživateli a stavu uživatelského účtu.

GeoServer použitý v systému jako implementace mapového serveru spravuje autorizaci a autentizaci ve vlastní režii, což v praxi znamená nutnost vytvářet uživatelské účty a jejich role obsažené v části nd-rest-backend a také v GeoServeru. V GeoServeru je však možné vytvořit nebo použít úložiště pro uživatelské účty z libovolného JDBC [54] připojení k databázi. Úložiště je reprezentované pevně definovaným schématem databázových tabulek. Schéma je odlišné od tabulek dostupných v databázi systému, tuto nekompatibilitu lze vyřešit vytvořením databázových pohledů z dostupné databáze, které simulují schéma pro GeoServer úložiště. Jediným problémem je přístupové heslo, jelikož se v databázi nachází v podobě haše složeného kromě hesla také z kryptografické soli. Algoritmy pro hašování v GeoServeru a Spring Security nejsou navzájem kompatibilní, uživatel se tudíž i přes existující propojení skrze úložiště nemůže autentizovat svým heslem. Problém je vyřešen následovně: hesla pro GeoServer jsou jen částí celkového haše, uživatel se neautorizuje přímo pomocí hesla, nýbrž zasláním předpočítané části haše. Jelikož není haš kompletní, jeho případný únik nezpůsobí bezpečnostní problém ve zbytku systému. Autorizace ke GeoServeru probíhá při každém HTTP požadavku pomocí HTTP autentizace. Uživatel má dle své role přiřazen přístup ke čtení a zápisu do poskytovaných mapových vrstev.

Část nd-client obsahuje přihlašovací formulář, jehož obsah je zasílán do části nd-rest-backend, která v případě správných autentizačních údajů vrátí seznam systémových rolí autentizovaného uživatele. Autorizovaný uživatel je poté po zbytek komunikace s částí nd-rest-backend identifikován pomocí `JSESSIONID` cookie. Kvůli přístupu ke GeoServeru je po úspěšném přihlášení spočítán z hesla a kryptografické soli částečný autentizační haš. Haš je trvale uložen spolu s uživatelským jménem a uživatelskými systémovými rolemi v rámci webového prohlížeče pomocí HTML5 funkcionality *local storage* [35]. Předpočítaný haš je následně spolu s uživatelským jménem používán v rámci požadavků na GeoServer. Knihovny klientské aplikace jsou přizpůsobeny spolu s celou klientskou aplikací budování obsahu aplikace dle systémových rolí, které jsou přiřazeny autentizovanému uživateli.

Vypršení autentizace je v části nd-client detekováno odpovědí z části nd-rest-backend nebo z GeoServeru se stavem 401. Po takto obdržené odpovědi jsou údaje uložené v local storage vynulovány a uživatel přesměrován na stránku s přihlašovacím formulářem.

Údaje přenášené z přihlašovacího formuláře do části nd-rest-backend je nutné stejně jako HTTP autentizační hlavičky použité při komunikaci s GeoServerem ochránit proti odposlechu. Zabránění odposlechu lze docílit pomocí HTTPS nadstavby HTTP protokolu, která přenášená data šifruje. Z těchto důvodů je veškerá realizovaná komunikace skrze HTTP realizována v aplikaci pomocí HTTPS.

Kapitola 7

Testování a nasazení

Při testování softwarového systému, který je architektonicky dekomponován na dílčí podčásti, je nutné testovat každou část zvlášť a posléze provést testování systému jako celku. Části systému, které nejsou předmětem jeho implementace (mapový server a prostorová databáze), jsou testovány dodavatelem daného softwaru, a proto obvykle není nutné je znovu testovat. Tato kapitola se zaměřuje na testování implementovaných částí systému, volbou nástrojů pro jejich testování a integračním testováním celého systému.

7.1 Testování nd-rest-backend

Testování části nd-rest-backend je nutné rozdělit na dva samostatné úkoly. První úkol se zaměřuje na testování pomocných utilit, tříd sloužících jako parametry REST rozhraní, tříd entit a obalovacích tříd. Testování bylo provedeno pomocí jednotkových (unit) testů. Jelikož je část nd-rest-backend implementována v platformě Java EE, pro ulehčení psaní jednotkových testů byl použit rámec *JUnit*.

Druhý z úkolů při testování části nd-rest-backend je poněkud obtížnější, jelikož realizuje testování EJB komponent vrstvy DAO a vrstvy komunikačního rozhraní ve spolupráci s prostorovou databází. EJB komponenty s použitím DI není možné testovat mimo EJB kontejner. Standardně se proto při testování EJB komponent setkáváme s vestavěným (embedded) EJB kontejnerem nebo přímo celým vestavěným Java EE aplikačním serverem. Kontejner nebo server je poté konfigurován a spouštěn při běhu jednotlivých testů. Změnou konfigurace lze například pro testovací prostředí zvolit jinou databázi a povolit její automatické generování, jak bylo popsáno v kapitole 6.3.4. Podobně jako v prvním úkolu bylo testování provedeno jednotkovými testy s rámcem JUnit. Navíc byla použita testovací platforma *Arquillian*, která mimo jiné poskytuje vestavěný EJB kontejner.

7.1.1 JUnit

JUnit [7] je nejčastěji používaným testovacím rámcem pro psaní jednotkových testů v aplikacích napsaných v programovacím jazyce Java [21]. Naprostá většina vývojových nástrojů pro tvorbu Java aplikací podporuje tvorbu a běh testů napsaných právě v tomto rámci. Jednotkový test (testovací případ) se v JUnit obvykle vytváří pro jednu třídu (jednotku) pomocí testovací třídy, která obsahuje testy pro jednotlivé metody jednotky. Metody testovací třídy jsou v JUnit 4, které je použito pro testování v implementovaném systému, opatřeny anotací `@Test`. Akce před a po inicializaci testovací třídy mohou být spouštěny ve statických metodách opatřených anotacemi `@BeforeClass` a `@AfterClass`. Před i po

každém testu se navíc vykonávají metody označené anotacemi `@Before` a `@After`. Uvedené metody obvykle slouží pro inicializaci a uvolnění zdrojů nutných k provedení testů. Obsah metod testovací třídy obvykle sestává z přípravy parametrů pro testovanou metodu, její volání a vyhodnocení kritérií, která má její výsledek splňovat. Kritéria jsou kontrolována pomocí statických metod `assert*` třídy `org.junit.Assert`. Příklad jednotkového testu je dostupný v kódu 7.1.

7.1.2 Arquillian

Arquillian je testovací platforma pro Java aplikace k tvorbě integračních, funkčních a akceptačních testů. Arquillian poskytuje běhové prostředí pro testy, kterým zprostředkovává služby kontejnerů, rámců a aplikačních serverů [53]. Arquillian se běžně používá spolu s rámcem JUnit 4. Jednotkový test využívající Arquillian je oproti běžnému JUnit testu rozšířen o definici části aplikace, která je v testu použita. Z této definice je vytvořen archív, který je spouštěn v prostředí Arquillian. V každém testu si tedy lze stanovit jednotlivé použité části archivu, což je užitečné například při záměnách konfiguračních souborů pro účely testování. Příklad jednotkového testu s definicí součástí archivu je dostupný v kódu 7.1 na řádcích 5 až 10.

```
1  @RunWith(Arquillian.class) // aktivace Arquillian běhového prostředí
2  public class LocationDaoImplTest {
3      @EJB private LocationDao dao; // DAO získané pomocí DI
4      private static WKTReader wktr; // staticky inicializovaný objekt
5      @Deployment public static JavaArchive createTestArchive() {
6          return ShrinkWrap.create(JavaArchive.class)
7                  .addPackage(Dao.class.getPackage())
8                  .addPackage(Location.class.getPackage())
9                  .addAsResource("test-persistence.xml");
10     } // <-- vytvoření archivu s testovací konfigurací DB, DAO a třídami entit
11     @BeforeClass public static void setupClass() throws ParseException {
12         wktr = new WKTReader(new GeometryFactory(new PrecisionModel()));
13     } // <-- statická inicializace při vytvoření testu
14     @Before public void setupData() throws ParseException {
15         assertNotNull(dao.create(new Location(wktr.read("POINT (1 2)"))));
16     } // <-- inicializace testovacích dat před každým testem
17     @After public void removeData() {
18         dao.removeAll();
19         assertEquals("not truncated", 0, dao.count());
20     } // <-- odstranění inicializovaných dat po každém testu
21     @Test public void testCount() {
22         assertEquals("invalid location count", 1, dao.count());
23     } // <-- test metody count pracující s daty přidanými při inicializaci
24 }
```

Kód 7.1: Příklad části jednotkového testu vytvořeného v JUnit rámci s použitím Arquillian platformy pro testování EJB komponenty z vrstvy DAO

V implementované aplikaci se platforma Arquillian používá při testování správnosti operací EJB komponent vrstvy DAO, jelikož jejich běh je podmíněn přítomností EJB kontejneru. V kódu 7.1 je možné pozorovat jednotkový test pro EJB komponentu spravující polohy z vrstvy entit. Vytvoření závislosti na EJB komponentě je patrná na řádku 3, vložení testovacích dat na řádcích 14 až 16 a jednoduchý test na řádcích 21 až 23.

7.2 Testování nd-client

V části nd-client je nutné testovat zejména klientské skripty. Jelikož výsledný systém je multiplatformní webová aplikace, musí testování probíhat v různých webových prohlížečích. Pro tvorbu jednotkových testů je použit rámec *Jasmine* a souběžné spouštění testů ve více webových prohlížečů je uskutečněno nástrojem *JsTestDriver*. Zbytek této kapitoly se věnuje popisu zmíněných technologií a způsobu jejich využití při testování části nd-client.

7.2.1 Jasmine

Jasmine je testovací rámec pro skriptovací jazyk JavaScript, koncipovaný pro použití při behaviour-driven agilním přístupu k vývoji webových aplikací [46]. Testy jsou organizovány do jednotek, podobně jako v případě JUnit. Jednotkový test se vytváří funkcí `describe`, které je předán název jednotky a funkce obsahující jednotlivé testy a jejich konfiguraci. Test je vytvořen voláním metody `it`, které je předáno jméno testu a tělo testu ve formě funkce. Tělo testu může obsahovat libovolný kód v jazyce JavaScript včetně volání Jasmine funkcí. Jedná se o funkce pro vyhodnocování testovaných hodnot, časování pro vyhodnocování asynchronních volání atd. Vyhodnocení testů probíhá otevřením testovací stránky v libovolném webovém prohlížeči. Z důvodů automatizace tohoto postupu byl pro běh a vyhodnocení testů použit JsTestDriver.

Testy napsané za pomoci rámce Jasmine se používají při testování nd-client především u utilit a méně rozsahlých komponent uživatelského rozhraní. Celková funkčnost je testována během integračního testování, které je popsáno v kapitole 7.3.

7.2.2 JsTestDriver

JsTestDriver je klient-server aplikace pro běh jednotkových testů v jazyce JavaScript [26]. Serverová část zprostředkovává připojení k libovolnému počtu webových prohlížečů, které jsou lokálně dostupné v místě jejího nasazení. Klientská část umožňuje vzdáleně zasílat požadavky na testování serverové části, které jsou formovány v konfiguračním souboru. Obsahem konfiguračního souboru jsou informace pro připojení k serverové části, lokální cesty k souborům s testy a k souborům s testovanými zdrojovými kódy. Serverová část zaslaný požadavek zpracuje, spustí jednotlivé testy ve všech připojených webových prohlížečích a informuje klienta o výsledku testů. Jelikož existuje pevná konfigurace, lze testy spouštět opakovaně a integrovat je do existujících nástrojů pro vývoj webových aplikací. Separace na klientskou a serverovou část umožňuje současně tvořit testy pro více serverů s různými variantami připojených webových prohlížečů. Díky separaci lze sdílet služby serverů pro testování více aplikací [49].

Testovací proces části nd-client využívá JsTestDriver pro zprostředkování běhového prostředí jednotkových testů, které jsou napsany za pomoci rámce Jasmine. K serverové části JsTestDriver jsou připojeny nejpoužívanější webové prohlížeče, čímž je do jisté míry zabezpečena multiplatformnost části nd-client.

7.3 Integrační testování

Integrační testování probíhá po nasazení všech dílčích podčástí systému do svých běhových prostředí s cílem otestovat jejich vzájemnou kooperaci a celkovou funkčnost systému. Pro implementovaný systém bylo zvoleno integrační testování pomocí automatického testovaní

funkcionality webové aplikace, které umožňuje opakovaně testovat zadané uživatelské interakce se systémem. Za tímto účelem byla použita sada nástrojů *Selenium*, které umožňují jednotlivé uživatelské interakce zaznamenávat a poté automaticky spouštět v různých prostředích.

7.3.1 Selenium

Selenium je sada nástrojů pro vzdálené ovládání webových prohlížečů používaná pro automatizované funkční testování webových aplikací [9]. Ovládání prohlížeče je realizováno pomocí sady dostupných příkazů, které umožňují na otevřené HTML stránce spouštět DOM [27] události, měnit obsah a strukturu DOM elementů stránky, čekat na splnění jistých podmínek, testovat platnost podmínek atd. Testovací případy složené z posloupnosti příkazů lze automaticky nahrávat z činnosti uživatele v prohlížeči pomocí nástroje *Selenium IDE*, který je dostupný pouze pro prohlížeč Firefox. Za účelem vytvoření testovacích případů pro jiné prohlížeče lze použít libovolný rámec pro tvorbu jednotkových testů a API dostupné z nástroje *Selenium Remote Control* nebo *Selenium WebDriver* [39]. Při použití těchto API je podobně jako v případě JsTestDriver zprostředkována funkcionalita lokálního nebo vzdáleného webového prohlížeče. Oba nástroje poskytují zpětnou vazbu pro kontrolu, zda jednotlivé kroky testů proběhly v cílovém webovém prohlížeči bez problémů.

Pro vytvoření Selenium integračních testů v implementovaném systému byly použity oba uvedené přístupy. Testovací sady vytvořené nahráním uživatelských akcí pomocí Selenium IDE byly posléze exportovány nástrojem obsaženým v Selenium IDE do jednotkových JUnit testů s využitím Selenium Remote Control. Tímto způsobem lze tvorbu testů značně urychlit. JUnit integrační testy jsou vhodnější než Selenium IDE testy, jelikož je možné je spouštět automaticky bez zásahu uživatele. Výhodou je také možnost sdílet části testů, které jsou často opakované. Jedná se například o část testu realizující přihlášení uživatele do systému. Nástroj Selenium Remote Control se stará o zpětnou vazbu

7.4 Nasazení

Všechny části vytvořeného GIS systému jsou s výjimkou prostorové PostGIS databáze implementovány pomocí Java EE platformy. Nasazení systému je tedy spojeno především s výběrem Java EE aplikačního serveru pro běh dílčích podčástí. Výhodou dekompozice systému je mimo jiné možnost nasadit jednotlivé části systému na dedikované stroje a tím zvýšit výkonnost systému. Některé části systému, jako například nd-client a mapový server, lze nasadit na více dedikovaných strojů a následně rozložit uživatelské požadavky na jednotlivé stroje, což může pomoci k obsluze většího počtu uživatelů. Obvykle je ale dostačující nasadit všechny části systému na jeden aplikační server a složitější způsob nasazení řešit až v případě nedostatečného výkonu systému.

Pro nasazení částí systému implementovaných v Java EE platformě byl zvolen aplikační server *Glassfish 4*, který poskytuje podporu pro Java EE platformu ve verzi 7 a obsahuje všechny technologie nutné pro běh jednotlivých částí systému.

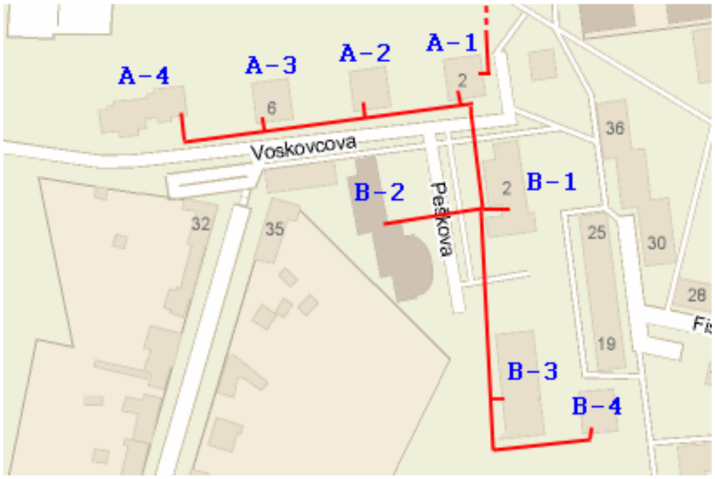
Pro distribuci implementovaných částí systému jsou použity soubory *WAR* [20], které jsou vytvořeny nástrojem *Maven* [12]. Nástroj Maven je použit v implementovaných částech systému pro jejich sestavení, definici softwarových závislostí, inicializaci testovacích procedur, běhových prostředí pro testy atd. Maven je široce rozšiřitelný díky existenci velkého množství zásuvných modulů. S ohledem na požadavek nízké datové náročnosti systému je výhodné použití modulu *yuicompressor* [4] pro minimalizaci zdrojových souborů v jazyku

JavaScript a souborů s CSS styly. Rozšíření je aplikováno během sestavení části nd-client a díky němu lze dosáhnout až 40% snížení velikosti jednotlivých souborů. Rozšíření rovněž umožňuje shlukovat více souborů do jednoho. Tato vlastnost je velmi užitečná, pokud vkládáme z důvodů lepší udržovatelnosti do jednoho souboru pouze jednu třídu jazyka JavaScript. Existence méně souborů umožňuje zrychlení načtení webové aplikace, jelikož běžný webový prohlížeč má omezený počet souběžných připojení k HTTP serveru a poskytnutí velkého počtu tříd po jednom souboru by trvalo delší dobu.

Kapitola 8

Případová studie

Případová studie pro testování vytvořeného GIS systému byla vybudována na projektové dokumentaci optické počítačové sítě z [59, s. 53–59]. Projektová dokumentace zahrnuje návrh pasivní optické sítě v ulicích Voskovcova a Peškova v katastru města Olomouce. Jedná se o menší přístupovou síť typu FTTH [32] pro 8 bytových domů. Přehled situace na mapě je dostupný na obrázku 8.1. Přístupová síť je rozvedena z jednoho centrálního bodu (A-1) do všech budov a dále do jednotlivých pater budov a bytů na příslušných patrech. Mimo budovy jsou prvky sítě vedené ve výkopech pod chodníky a chráničkách. Ve dvou místech je učiněn průtlak pod existující vozovkou. Do každé budovy je dovedeno sedm mikrotrubiček, přičemž pouze jedna obsahuje optický kabel s šesti vlákny a ostatní slouží jako záloha. Křížení mikrotrubiček je realizováno pomocí Y spojek. Do centrálního prvku jsou přivedeny dva optické kabely o dvanácti vláknech. Dvě vlákna z každého kabelu vedoucího k jednotlivým budovám jsou navařena na vlákna přivedených kabelů. Centrálním prvkem je optický rozvaděč, který obsahuje uložení pro sváry. V každé budově je na jedno z přivedených vláken instalován splitter s rozbočovacím poměrem 1:32 nebo 1:64. Ostatní přivedená vlákna jsou ponechána jako záloha. Rozvody po budově nejsou dále předmětem případové studie.



Obrázek 8.1: Přehledová situace přístupové sítě použité pro případovou studii [59]

Výše stručně popsaná počítačová síť byla zvolena pro potřeby případové studie, jelikož zadavatel kladl důraz na umožnění pasportizace podobných pasivních optických přístupových sítí.

8.1 Postup pasportizace případové studie pomocí vytvořeného GIS systému

Převod projektové dokumentace použité pro účely případové studie probíhal ve vytvořeném systému v následujících krocích:

1. zaznamenání výkopů mezi budovami včetně jejich poloh pomocí tras s typem „výkop“,
2. zaznamenání chráničeк nebo fólií umístěných ve výkopech mezi jednotlivé uzly sítě,
3. zaznamenání uzlů sítě (Y-spojky) pomocí kabelových komor s příslušným typem,
4. zaznamenání dvou přívodních kabelů s příslušnými počty vláken vedoucích do centrálního bodu,
5. zaznamenání optického rozvaděče v centrálním bodě,
6. zaznamenání mikrotrubiček v chráničkách pro vedení kabelů z centrálního bodu do budov,
7. zaznamenání kabelů s šesti vlákny vedených z centrálního bodu do každé budovy,
8. zaznamenání svárů mezi vlákny přívodních kabelů a kabelů přístupové sítě v optickém rozvaděči umístěném v centrálním bodě,
9. zaznamenání pasivních optických rozbočovačů v každé budově a jejich spojení s jedním z vláken přivedených do budovy,
10. zaznamenání adresních bodů jednotlivých budov a svázání adresních bodů s polohou rozbočovačů.

8.2 Vyhodnocení případové studie

V ukázaném postupu pasportizace případové studie je patrné, že se vytvořený systém snaží v návaznosti jednotlivých kroků simulovat fyzický proces výstavby sítě. Tato analogie zvyšuje intuitivnost při práci koncového uživatele se systémem.

Úspěšně provedená pasportizace pasivní optické sítě ukazuje, že primární funkce systému je funkční. Externí původ případové studie vylučuje její účelové vytvoření tak, aby byla systémem snadno proveditelná.

Kapitola 9

Závěr

Cílem práce bylo vytvořit geografický informační systém určený k pasportizaci a vizualizaci rozlehlých počítačových sítí se zaměřením na jejich fyzickou strukturu. Před započetím analýzy systému se práce věnovala úvodu do problematiky GIS systémů a okrajově se zabývala prostorovými databázovými systémy, které jsou s GIS úzce spjaty. Po zahájení prací na analýze systému se pro její úspěšné zpracování ukázalo nezbytné identifikovat prvky počítačových sítí, které má smysl pasportizovat. U pasportizace optických počítačových sítí byla navíc vyhodnocena nutnost pasportizovat součásti jejich tras. Ze shromážděných informací byl zhotoven návrh formátu pasportu, určen způsob zaznamenání geografických poloh sledovaných prvků a nastíněn převod formátu do softwarové podoby. Návrh formátu pasportu byl konzultován se zadavatelem. Vzniklé námitky ze strany zadavatele byly do výsledného formátu promítnuty a následně byl formát zadavatelem schválen. Po dokončení formátu pasportu byly ve spolupráci se zadavatelem specifikovány a analyzovány funkční požadavky na systém. Porovnáním stanovených požadavků s vlastnostmi existujících řešení byla vyhodnocena nutnost vyvinout vlastní řešení pro splnění všech kladených požadavků. Po dokončení analýzy systému byl stanoven předběžný plán etap projektu a naplánovány pravidelné schůzky určené k ověření splnění cílů etapy, konzultaci vzniklých problémů při řešení etapy a rozvinutí cílů následující etapy. První krok návrhu systému demonstroval zadavateli stěžejní technologie vybrané k realizaci systému a jejich význam v celkové architektuře sytému. Jednalo se zejména o knihovnu OpenLayers pro tvorbu interaktivních map, mapový server (GeoServer), prostorovou databázi (PostGIS) a platformu Java EE. Raný výběr a schválení technologií pro realizaci systému zabránil budoucím změnám architektury, které komplikují vývoj některých softwarových produktů. Pokračování návrhu systému sestávalo z dekompozice implementovaných částí systému, které byly stanoveny v architektuře systému, návrhu uživatelského rozhraní a návrhu struktury dat ze stanoveného formátu pasportu. Při vypracování návrhu bylo dbáno na rozšiřitelnost systému a mimo jiné bylo navrženo budoucí možné napojení na externí softwarové systémy. Navržený systém byl implementován a průběžně konzultován se zadavatelem. Během implementace také postupně vznikaly testy pro ověření správnosti kritických funkcí a operací. Testovaní bylo završeno integračním testováním pomocí sady nástrojů Selenium. Pro ověření fungování primární funkce systému (evidence a vizualizace pasportu) byla zhotovena případová studie, která ukázala schopnost systému pasportizovat reálnou pasivní optickou přístupovou síť. Po dokončení implementace byly uvedeny možné způsoby nasazení systému do produkčního prostředí.

Hlavním přínosem práce je samotný vytvořený GIS systém, který je určen primárně pro pasportizaci optických počítačových sítí zadavatele. Po osvědčení systému v síti zadavatele

je plánováno jeho možné budoucí poskytnutí jiným odběratelům. Dalším stěžejním přínosem práce je definice formátu softwarového pasportu pro pasportizaci fyzické struktury počítačových sítí, jelikož dle vědomosti autora nebyl podobný formát zatím publikován. Celý text bude mimo jiné sloužit pro zaškolení nových vývojářů, kteří budou na rozvoji systému pracovat v budoucnosti.

Během řešení práce vzniklo několik problémů. Při analýze a návrhu bylo nutné se vyrovnat s nezkušeností zadavatele s podobnými typy projektů, jelikož jeho primární činností je poskytování internetového připojení třetím osobám. Naopak orientace zadavatele v oboru počítačových sítí byla prospěšná při zhotovení formátu pasportu. Zmíněný problém byl vyřešen častými schůzkami a telekonferencemi s pověřenými pracovníky zadavatele, na kterých byly dílčí výsledky a postupy analýzy a návrhu systému osvětleny a diskutovány. I přes zavedený postup byly v implementační části objeveny chyby v návrhu, týkající se především drobných úprav struktury dat. Objevené chyby ale díky dekompozici systému a obecnosti používaných komponent neměly za následek významné pozdržení prací. Problémy během implementace vznikaly především z důvodů nepřizpůsobitelnosti některých použitých nástrojů, knihoven a rámců tak, aby vyhovovaly požadavkům kladeným na systém. Jelikož byly pro implementaci vybrány nástroje, knihovny a rámce s otevřenými zdrojovými kódy, ve většině případech nebylo obtížné do nich zanést změny nutné k přizpůsobení jejich vlastností potřebám systému. Posledním problémem vzniklým během řešení práce bylo nalézt vhodný způsob a technologie pro testování vznikajícího systému. Nakonec bylo zvoleno řešení, které stanovuje způsob a technologie testování pro jednotlivé dekomponované části sytému a integračního testování pro testování funkčnosti celého systému. Zvolený způsob by měl v budoucnu napomoci při rozdělení vývojového týmu do více skupin dle podčástí systému.

GIS systém vytvořený v rámci práce splňuje kritéria, která na něj zadavatel klade. V mnoha ohledech jej ale lze vylepšit, a proto je na konec práce umístěna kapitola, která se věnuje návrhu možných vylepšení systému.

9.1 Návrhy na budoucí vylepšení

Vytvořený GIS systém je vhodné rozšířit funkcionalitou uvedenou v následujícím seznamu:

- schématický editor pro editaci logické návaznosti síťových prvků,
- realizace napojení na systém FreenetIS definované v kapitole 5.2.4,
- použití hierarchického modelu pro uložení pozic prvků sítě,
- automatizovaný import výkopů z výkresu v CAD formátu,
- přizpůsobení uživatelského rozhraní pro mobilní zařízení s operačními systémy Android a iOS,
- tvorba uživatelské dokumentace,
- migrace implementace interaktivní mapy na knihovnu OpenLayers 3,[1]
- rozšíření o analytické funkce k plánování výstavby sítí,
- vkládání příloh ve formě souborů k některým zaznamenaným prvkům sítě,

[1] V době dokončení diplomové práce nebyla finální verze knihovny OpenLayers 3 vydána.

- tvorba dalších testovacích sad,
- oddělení statických dokumentů a JSP stránek v části nd-client tak, aby bylo možné poskytovat statický obsah běžným webovým serverem.

Rozšíření nebyla provedena v aktuální verzi systému především kvůli jejich časové náročnosti. Valná většina představených rozšíření je již naplánována k budoucímu zpracování.

Literatura

[1] Advance Fiber Optics, Inc.: OSPInSight Software [online]. 2014 [cit. 2014-01-02]. Dostupné z: http://ospinsight.com/content/?page=41.

[2] Aime, A.; Deoliveira, J.: Comparing the Performance of Open Source Web Map Servers [online]. 2008 [cit. 2014-03-12]. Dostupné z: http://presentations.opengeo.org/2008_FOSS4G/WebMapServerPerformance-FOSS4G2008.pdf.

[3] Aitchison, A.: *Pro Spatial with SQL Server 2012*. Apressus Series, Apress, 2012, 560 s., ISBN 9781430234913.

[4] Alchim31: YUI Compressor Maven Mojo [online]. 2010 [cit. 2014-05-11]. Dostupné z: http://alchim.sourceforge.net/yuicompressor-maven-plugin/index.html.

[5] Ambrož, J.: *Návrh, výstavba a měření optických přístupových sítí*. Diplomová práce, Vysoké učení technické v Brně, Brno, 2011.

[6] Barnet, J.: *Určení přesnosti transformace souřadnic pro výzkum odchylek od ideální trajektorie vozidla*. Diplomová práce, ČVÚT, Praha, 2008.

[7] Beck, K.: *JUnit Pocket Guide*. O'Reilly Media, 2004, 62 s., ISBN 05-960-0743-4.

[8] Brady Worldwide, Inc.: NetDoc®2 Cable Management Software Features [online]. 2014 [cit. 2014-01-02]. Dostupné z: http://www.bradyid.com/bradyid/cms/contentView.do/6590/gp.html.

[9] Burns, D.: *Selenium 2 Testing Tools: Beginner's Guide*. Packt Publishing, 2012, 437 s., ISBN 9781849518314.

[10] Břehovský, M.; Jedlička, K.; Šíma, J.: *Úvod do Geografických Informačních Systémů*. Západočeská Univerzita, 2003, 116 s.

[11] Češelský, J.: *Pasportizace v kontextu udržitelného managementu obecního domovního a bytového fondu*. VŠB – Technická univerzita, Fakulta stavební, 2011, s. 10–13, ISBN 978-80-248-2549-6.

[12] Company, S.: *Maven: The Definitive Guide: The Definitive Guide*. O'Reilly Media, 2008, 470 s., ISBN 978-059-6551-780.

[13] Deoliveira, J.: Introducing GeoDB [online]. 2014 [cit. 2014-05-15]. Dostupné z: https://github.com/jdeolive/geodb.

[14] EMCORE Corporation: Fiber Optic Components [online]. 2013 [cit. 2013-12-28]. Dostupné z: http://www.fiber-optics.info/articles/components/.

[15] Freeman, A.: *Pro jQuery*. Apressus Series, Apress, 2012, 1016 s., ISBN 978-143-0240-952.

[16] FreenetIS: Informační systém pro neziskové sítě [online]. 2014 [cit. 2014-01-12]. Dostupné z: http://www.freenetis.org/.

[17] Geary, D.: *Core Jstl: Mastering the Jsp Standard Tag Library*. Java 2 Platform, Enterprise Edition Series, Prentice Hall PTR, 2003, 584 s., ISBN 01-310-0153-1.

[18] GISOFT, v.o.s.: SPIDER-Fiber – tvorba a správa dokumentace optických sítí [online]. 2014 [cit. 2014-01-02]. Dostupné z: http://www.gisoft.cz/SPIDER-Fiber/SPIDER-Fiber.

[19] Golden, P.; Dedieu, H.; Jacobsen, K.: *Fundamentals of DSL Technology*. Taylor & Francis, 2004, 384 s., ISBN 978-020-3317-495.

[20] Gupta, A.: *Java EE 6 Pocket Guide*. O'Reilly Media, 2012, ISBN 978-1-449-33668-4.

[21] Harold, E. R.: An early look at JUnit 4 [online]. 2005 [cit. 2014-03-10]. Dostupné z: http://www.ibm.com/developerworks/java/library/j-junit4/.

[22] Hazzard, E.: *OpenLayers 2.10 Beginner's Guide*. Packt Publishing, Limited, 2011, 372 s., ISBN 9781849514132.

[23] Holzner, S.: *Ajax Bible*. Bible, Wiley, 2008, 695 s., ISBN 978-047-0377-512.

[24] Hrubý, M.: *Geografické Informační Systémy*. Fakulta informačních technologií, VUT v Brně, 2006, 82 s.

[25] JO Software Engineering GmbH: The Network Management Software cableScout [online]. [cit. 2014-01-01]. Dostupné z: http://www.josoftware.com/en/cable-scout/.

[26] Jurčovičová, M.: JavaScript Testing with JSTestDriver [online]. 2012 [cit. 2014-04-28]. Dostupné z: http://meri-stuff.blogspot.cz/2012/01/javascript-testing-with-jstestdriver.html.

[27] Keith, J.; Sambells, J.: *DOM Scripting: Web Design with JavaScript and the Document Object Model*. 2010, 336 s.

[28] Kolář, D.: *Pokročilé databázové systémy – prostorové databáze*. Fakulta informačních technologií, VUT v Brně, 2006, s. 14–15.

[29] Kothuri, R.; Godfrind, A.; Beinat, E.: *Pro Oracle Spatial for Oracle Database 11g*. Expert's Voice in Oracle, Apress, 2007, 372 s., ISBN 978-143-0204-466.

[30] Kratochvíla, J.: *Systém pro správu lokální sítě a služeb ISP*. Brno, 2012. 54 s. Bakalářská práce. Masarykova univerzita, Fakulta informatiky.

[31] Kropla, B.: *Beginning MapServer: Open Source GIS Development*. Expert's Voice in Open Source, Apress, 2005, 447 s., ISBN 9781430200536.

[32] Lafata, P.; Vodrážka, J.: Pasivní optická síť GPON [online]. 2009 [cit. 2013-12-30]. Dostupné z: http://access.feld.cvut.cz/view.php?nazevclanku=pasivni-opticka-sit-gpon&cisloclanku=2009050002.

[33] Laudon, K.; Laudon, J.: *Management Information Systems, Global Edition*. Pearson Education, Limited, 2013, 65 s., ISBN 02-737-8997-X.

[34] Linwood, J.; Minter, D.: *Beginning Hibernate*. Apresspod Series, Apress, 2010, 400 s., ISBN 978-143-0228-516.

[35] Lubbers, P.; Salim, F.; Albers, B.: *Pro HTML5 Programming*. Expert's voice in Web development, Apress, 2011, 352 s., ISBN 9781430238652.

[36] Maesen, K.: Hibernate Spatial: Overview [online]. 2014 [cit. 2014-05-13]. Dostupné z: http://www.hibernatespatial.org/.

[37] Maling, D. H.: Coordinate Systems and Map Projections for GIS, 2004, Dostupné z: http://www.spatial.maine.edu/~beard/Lectures510/Projections04.pdf.

[38] Mills, C.: *Practical CSS3: Develop and Design*. Develop and design, Peachpit Press, 2013, 320 s., ISBN 03-218-2372-9.

[39] Neff, P.: JavaScript testing [online]. 2010 [cit. 2014-04-26]. Dostupné z: http://weblog.patrice.ch/2010/02/10/javascript-testing.html.

[40] Open Geospatial Consortium: Geography Markup Language [online]. 2014 [cit. 2014-05-08]. Dostupné z: http://www.opengeospatial.org/standards/gml.

[41] Open Geospatial Consortium: Keyhole Markup Language [online]. 2014 [cit. 2014-05-08]. Dostupné z: http://www.opengeospatial.org/standards/kml.

[42] Open Geospatial Consortium: Web Feature Service [online]. 2014 [cit. 2014-05-10]. Dostupné z: http://www.opengeospatial.org/standards/wfs.

[43] Open Geospatial Consortium: Web Map Service [online]. 2014 [cit. 2014-05-10]. Dostupné z: http://www.opengeospatial.org/standards/wms.

[44] Open Geospatial Consortium: OpenGIS Simple Features Specification For SQL – Revision 1.1. OpenGIS Project Document 99-049, 2008-05-05.

[45] Oracle Corporation: Jersey: RESTful Web Services in Java [online]. 2014 [cit. 2014-05-15]. Dostupné z: https://jersey.java.net/.

[46] Pivotal Labs: Jasmine [online]. 2014 [cit. 2014-04-25]. Dostupné z: http://jasmine.github.io/2.0/introduction.html.

[47] Pokorný, J.: *Prostorové objekty a SQL. In GIS Ostrava 2001. Sborník konference*. Ostrava: VŠB - TUO, 2001, ISSN 1213-239X.

[48] PostGIS Developers: PostGIS – PostGIS Feature List [online]. 2014 [cit. 2014-03-12]. Dostupné z: http://postgis.net/features/.

[49] Prachař, J.: Automatizované testování JavaScriptu [online]. 2012 [cit. 2014-04-26]. Dostupné z: http://www.zdrojak.cz/clanky/automatizovane-testovani-javascriptu/.

[50] Pražák, J.: Slovník VÚGTK: geografická data, geodata, geoprostorová data [online]. [cit. 2013-12-01]. Dostupné z: http://www.vugtk.cz/slovnik/1070_geograficka-data--geodata--geoprostorova-data.

[51] Prochazka, D.: Mapové servery: Architektura a komunikační standarty, 2012-05-02, Dostupné z: http://perchta.fit.vutbr.cz/vyuka-gis/uploads/1/mapove_servery_compact.1.pdf.

[52] Ramsey, P.; Winslow, D.; Garnett, J.: Introduction to PostGIS: Introduction [online]. 2011 [cit. 2013-12-15]. Dostupné z: http://workshops.boundlessgeo.com/postgis-intro/introduction.html.

[53] Red Hat, Inc.: Arquillian Invasion! [online]. 2014 [cit. 2014-04-24]. Dostupné z: http://arquillian.org/invasion/.

[54] Reese, G.: *Database Programming with JDBC and Java*. Java (o'Reilly) Series, O'Reilly, 2000, 328 s., ISBN 15-659-2616-1.

[55] Roebuck, K.: *Object-Relational Mapping: High-Impact Strategies – What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors*. Emereo Pty Limited, 2011, 634 s., ISBN 978-174-3044-759.

[56] Schneider, R. D.: *MySQL: oficiální průvodce tvorbou, správou a laděním databází*. Grada, 2006, 372 s., ISBN 80-247-1516-3.

[57] Sinclair, I.: *Passive Components for Circuit Design*. Elsevier Science, 2000, s. 5–7, ISBN 978-008-0513-591.

[58] Sun Microsystems, Inc.: Core J2EE Patterns - Data Access Object [online]. 2002 [cit. 2014-05-14]. Dostupné z: http://www.oracle.com/technetwork/java/dataaccessobject-138824.html.

[59] Tejkal, V.: *Návrh optické přístupové sítě FTTx*. Brno, 2009. 81 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií.

[60] Trulove, J.: *Sítě LAN*. Grada, 2009, 384 s., ISBN 978-80-247-2098-2.

[61] Vojtičko, A.: *Terminologický slovník geodézie, kartografie a katastra*. ÚGKK, 1998, 540 s.

[62] Voženílek, V.: *Geografické Informační Systémy: Pojetí, historie, základní komponenty. I.* Vydavatelství Univerzity Palackého, 1998, 173 s., ISBN 80-7067-802-X.

[63] W3Techs: Usage of JavaScript libraries for websites [online]. 2014 [cit. 2014-04-13]. Dostupné z: http://w3techs.com/technologies/overview/javascript_library/all.

[64] Webber, J.; Parastatidis, S.; Robinson, I.: *REST in Practice: Hypermedia and Systems Architecture*. Theory in practice series, O'Reilly Media, 2010, 448 s., ISBN 978-144-9396-923.

[65] Winch, R.: *Spring Security 3.1*. Community experience distilled, Packt Publishing, Limited, 2012, 456 s., ISBN 9781849518277.

[66] World Wide Web Consortium: SOAP Version 1.2 Part 1: Messaging Framework [online]. 2007 [cit. 2014-04-10]. Dostupné z: http://www.w3.org/TR/soap12-part1/.

[67] Youngblood, B.: *GeoServer Beginner's Guide*. Community experience distilled, Packt Publishing, Limited, 2013, 320 s., ISBN 9781849516693.

Seznam zkratek

AJAX	Asynchronous JavaScript and XML
CAD	Computer aided design
CGI	Common Gateway Interface
CSS	Cascading Style Sheet
ČÚZK	Český úřad zeměměřický a katastrální
DAO	Data Access Object
DI	Dependency injection
DSL	Digital Subscriber Line
DXF	AutoCAD Drawing Exchange Format
EJB	Enterprise Java Beans
GIS	Geografický informační systém
GML	Geography Markup Language
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
Java EE	Java Enterprise Edition
JAX-RS	Java API for RESTful Web Services
JAXB	Java Architecture for XML Binding
JNDI	Java Naming and Directory Interface
JPQL	Java Persistence Query Language
JSON	JavaScript Object Notation
JSP	JavaServer Pages

JSP	JavaServer Pages
JSTL	JavaServer Pages Standard Tag Library
JVM	Java Virtual Machine
KML	Keyhole Markup Language
LAN	Local Area Network
MAN	Metropolitan Area Network
OAN	Active optical network
OGC	Open Geospatial Consortium
PHP	Hypertext Processor
POJO	Plain Old Java Object
PON	Passive optical network
PTM	Point to Multipoint
PTP	Point to Point
REST	Representational State Transfer
S-JTSK	Souřadnicový systém jednotné trigonometrické sítě katastrální
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WAN	Wide Area Network
WAR	Web Application Archive
WCS	Web Coverage Service
WFS	Web Feature Service
WFS	Web Map Service
WGS-84	World Geodetic System 1984

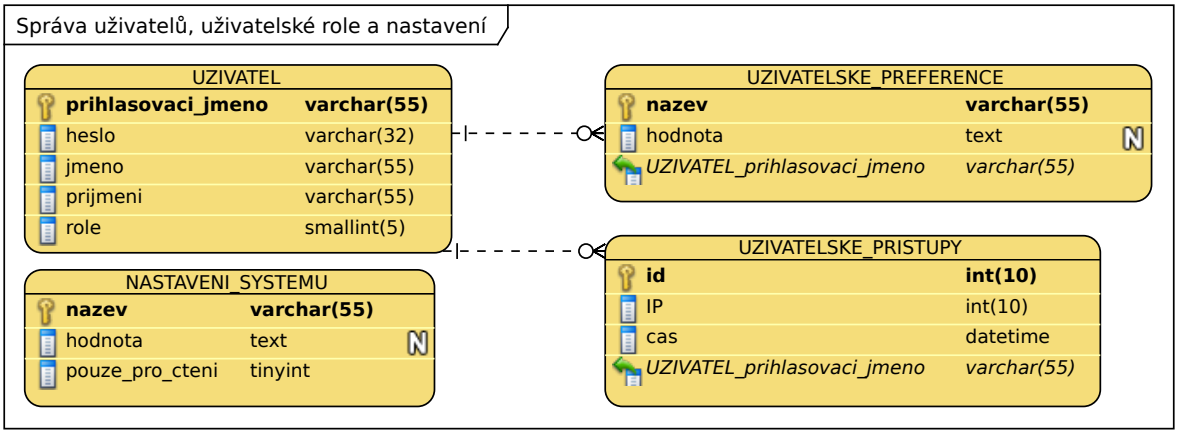
Příloha A

Obsah přiloženého CD

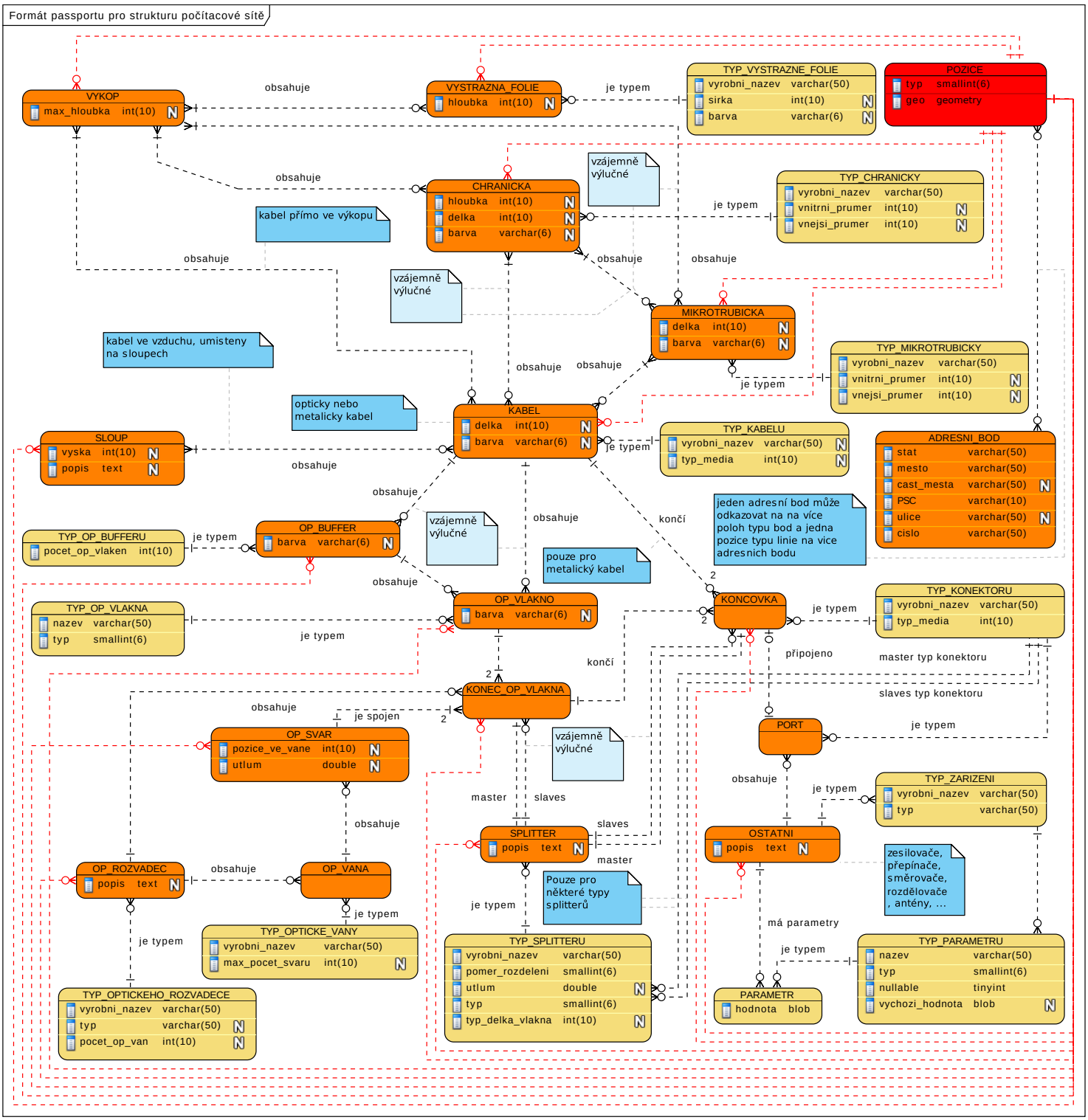
Cesta	Popis
`src`	adresář se zdrojovými kódy systému
`src/nd-client`	adresář se zdrojovými kódy části systému `nd-client`
`src/nd-rest-backend`	adresář se zdrojovými kódy části systému `nd-rest-backend`
`itest`	adresář s ukázkovým integračním testem pro Selenium IDE
`src-text`	adresář se zdrojovými tvarem technické zprávy ve formátu LaTeX

Příloha B

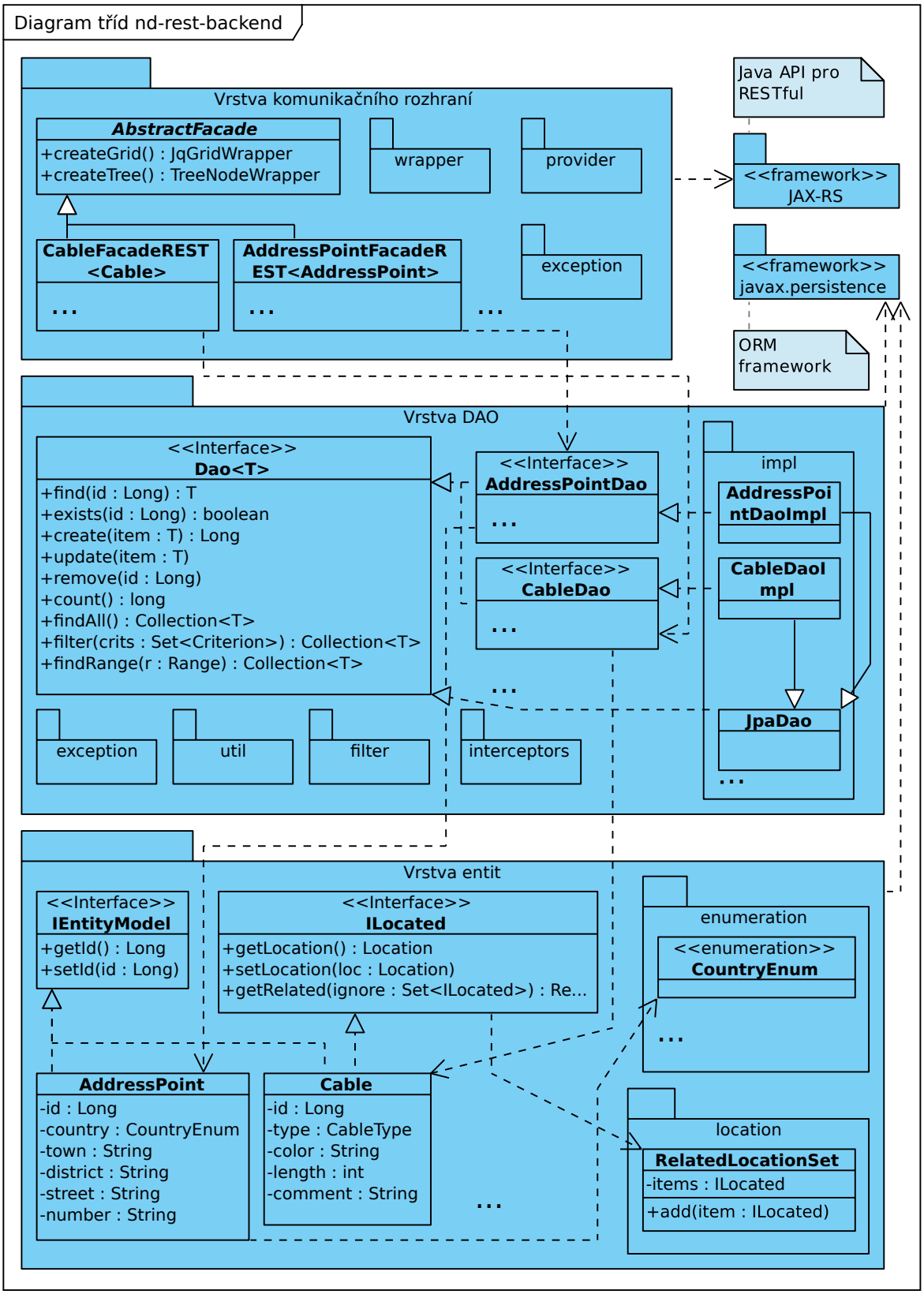
Návrhové diagramy



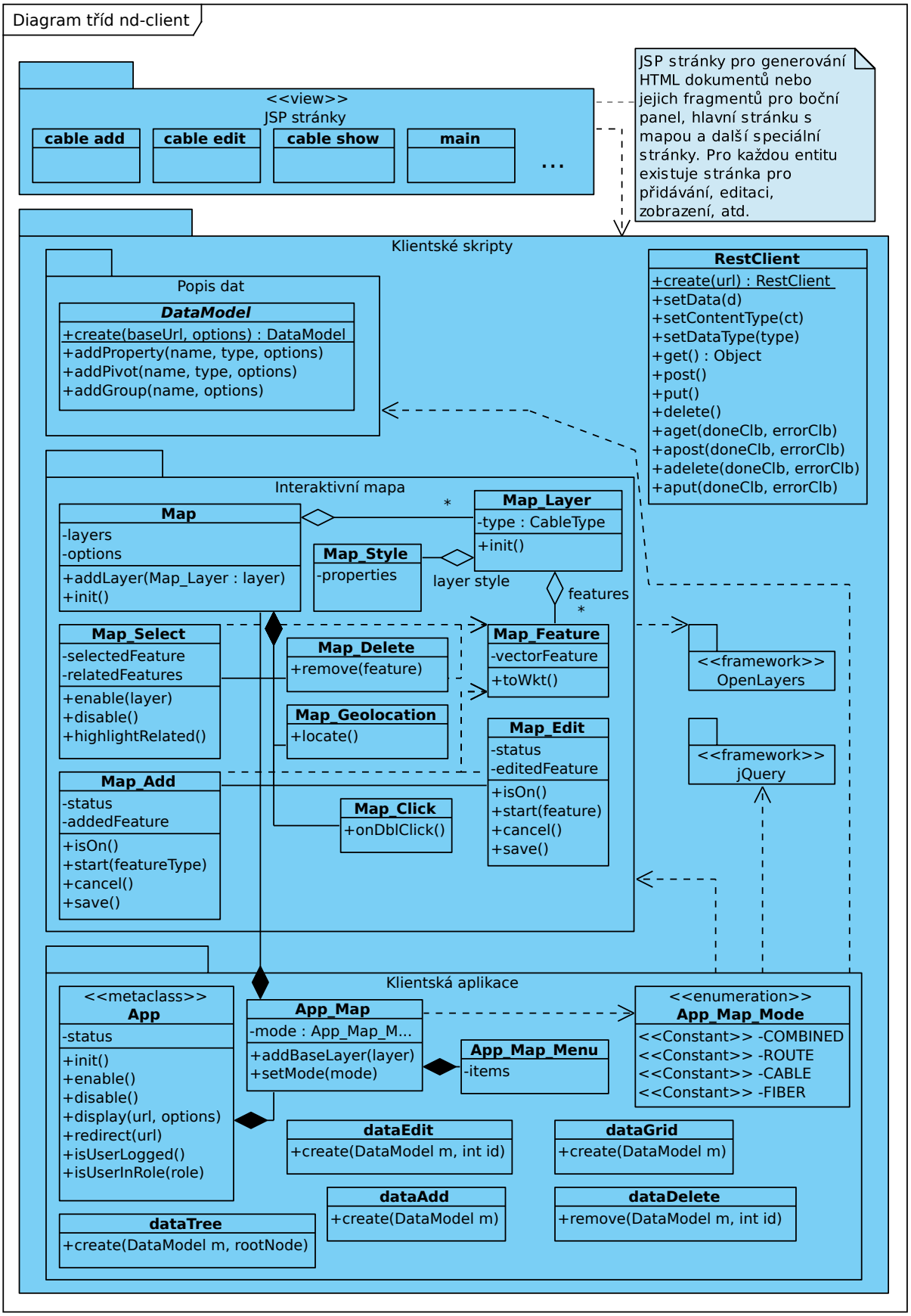
Obrázek B.1: Struktura dat pro správu uživatelských účtů, uživatelských rolí a nastavení ve formě ER diagramu



Obrázek B.2: Formát pasportu pro strukturu počítačové sítě ve formě ER diagramu



Obrázek B.3: Zjednodušený diagram tříd jazyka UML části systému nd-rest-backend



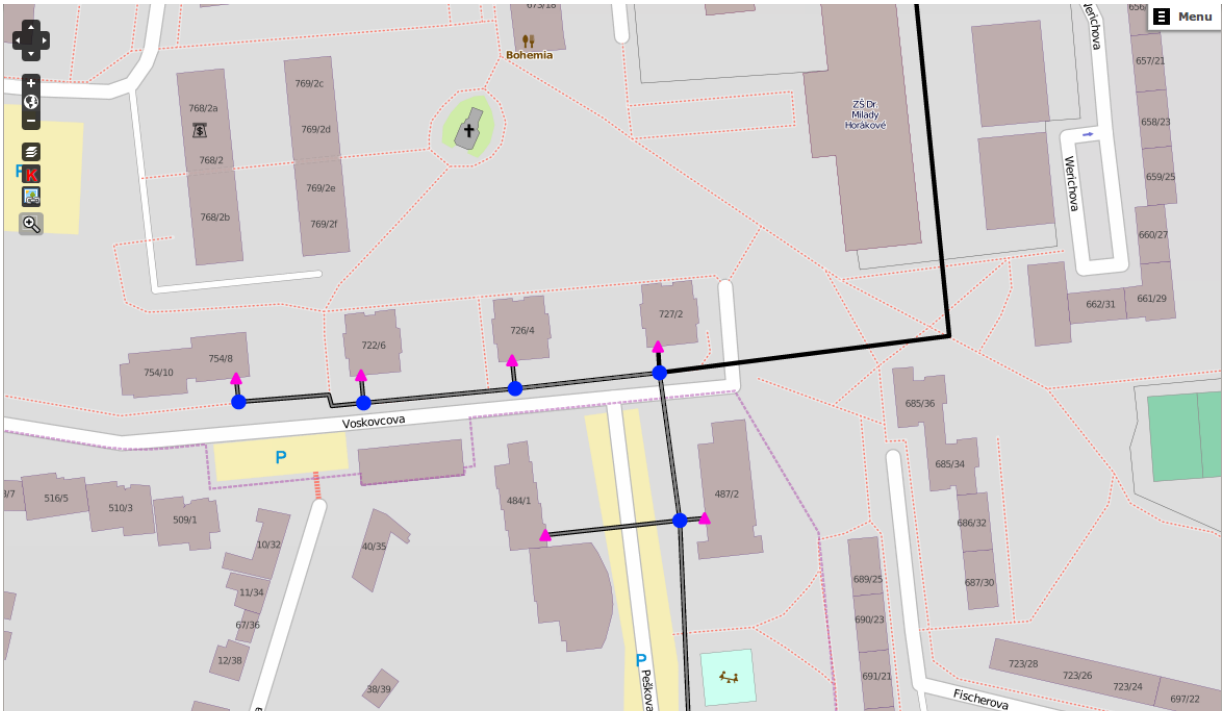
Obrázek B.4: Zjednodušený diagram tříd jazyka UML části systému nd-client

Příloha C

Snímky obrazovek systému

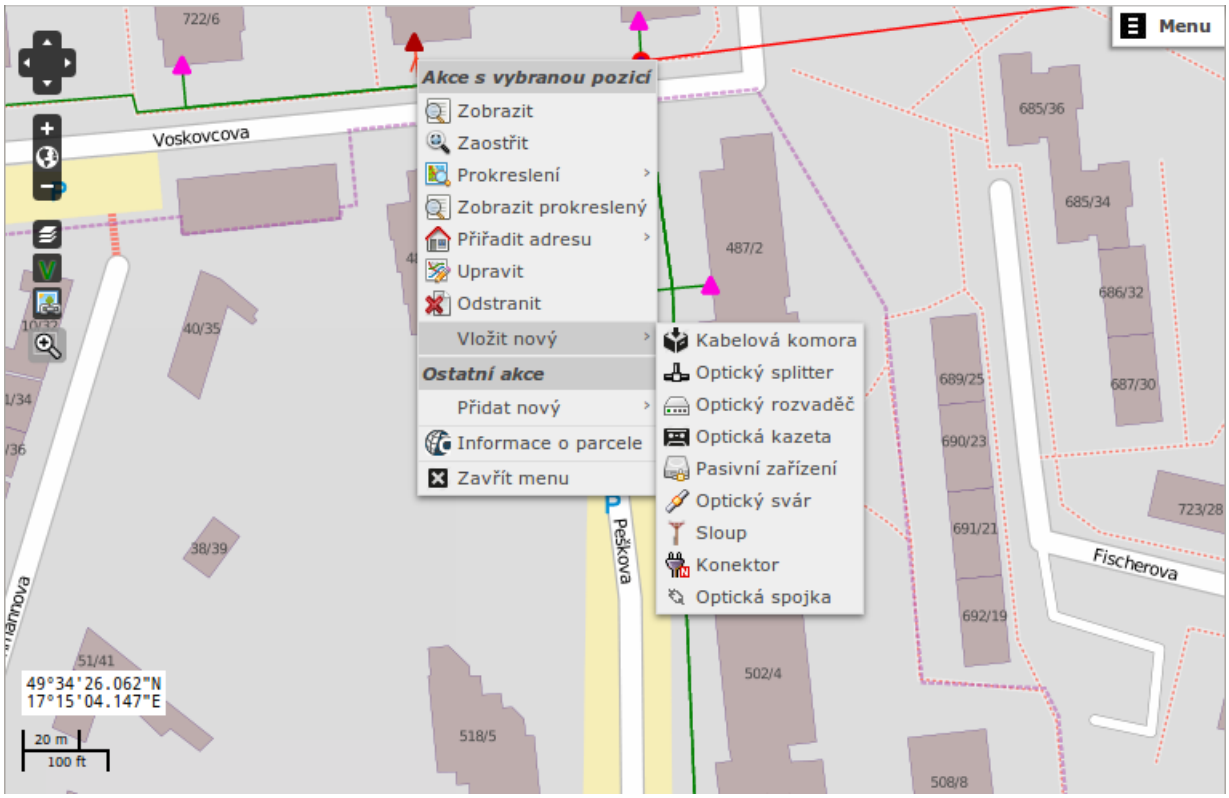
Snímky obrazovek jsou pořízeny v systému obsahujícím data z případové studie z kapitoly 8.

Obrázek C.1: Přihlašovací obrazovka

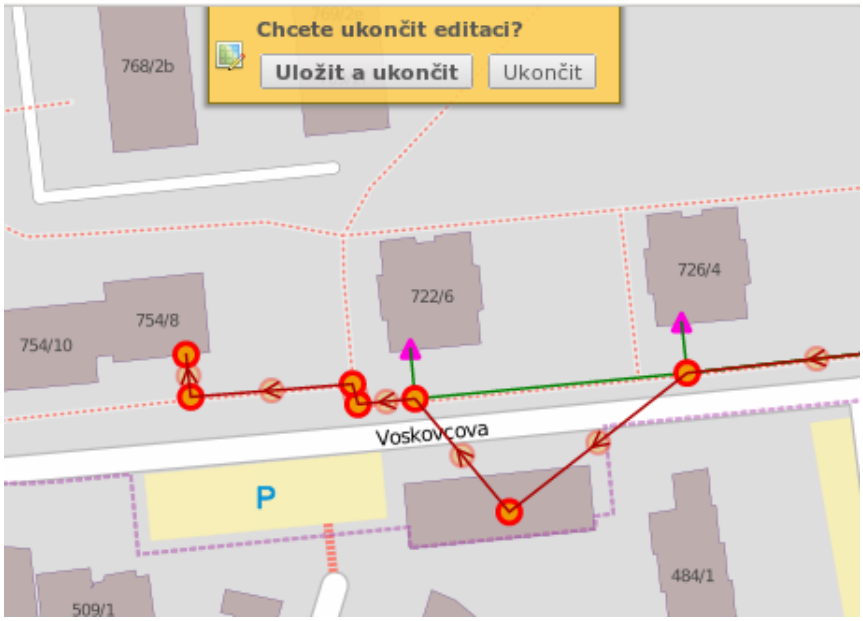


Obrázek C.2: Pohled na interaktivní mapu v kombinovaném módu s podkladovou vrstvou OpenStreetMaps

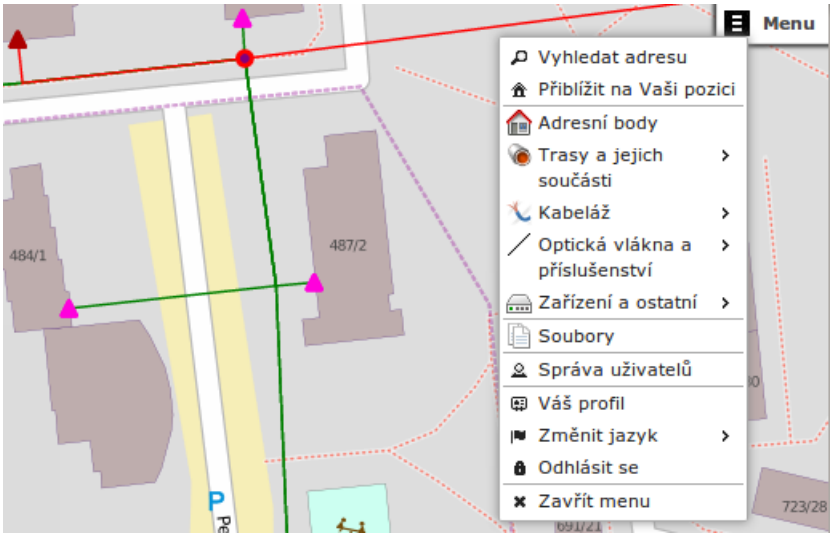
Obrázek C.3: Pohled na interaktivní mapu v módu optických vláken s podkladovou vrstvou Bing Aerial



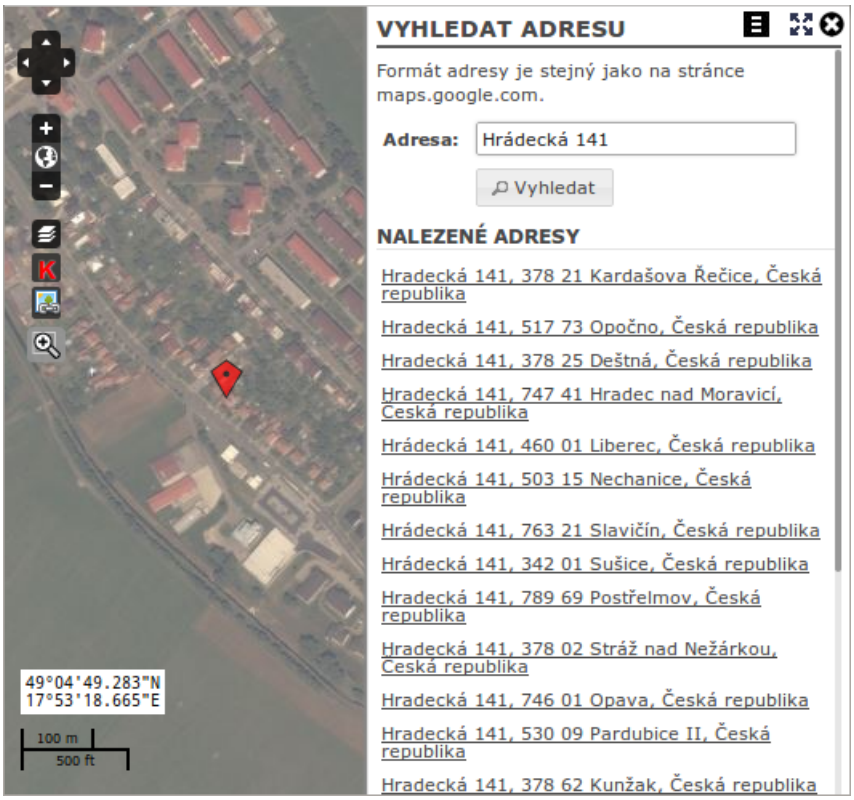
Obrázek C.4: Interaktivní mapa v módu optických vláken s otevřenou kontextovou nabídkou



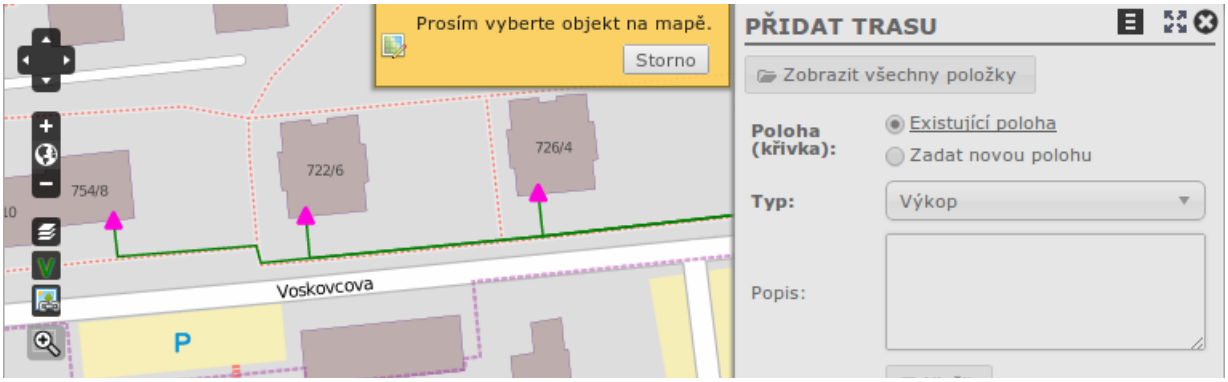
Obrázek C.5: Editace objektu mapy



Obrázek C.6: Hlavní nabídka klientské aplikace

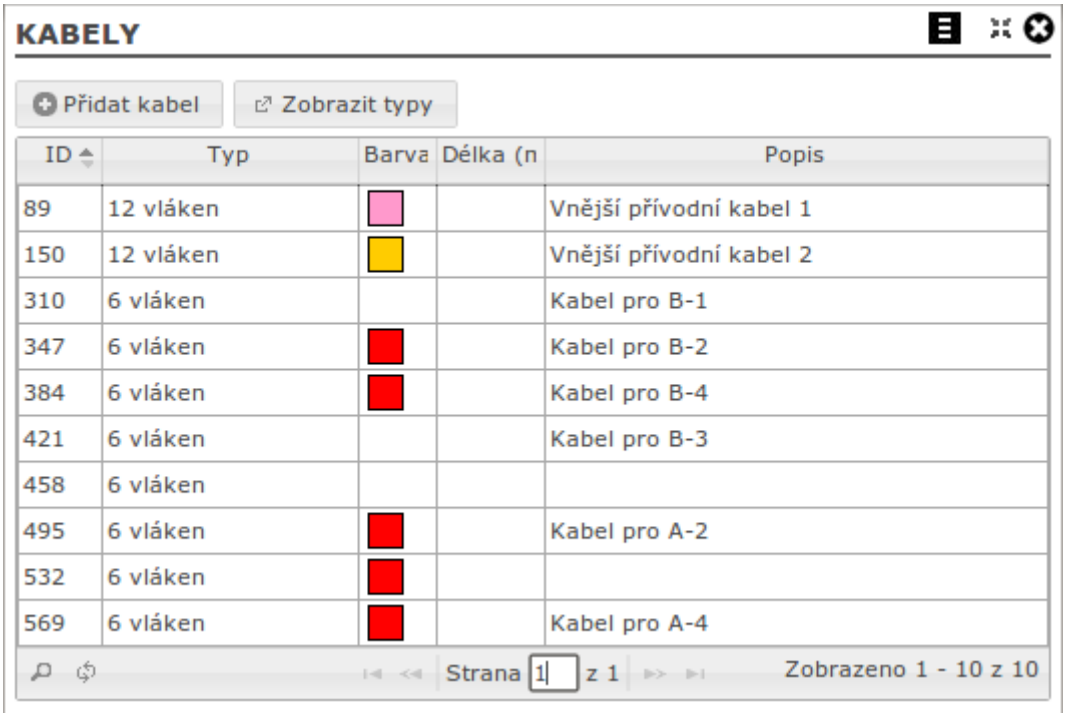


Obrázek C.7: Zobrazená klientská aplikace s formulářem pro vyhledání adresy a její následné vyznačení na mapě

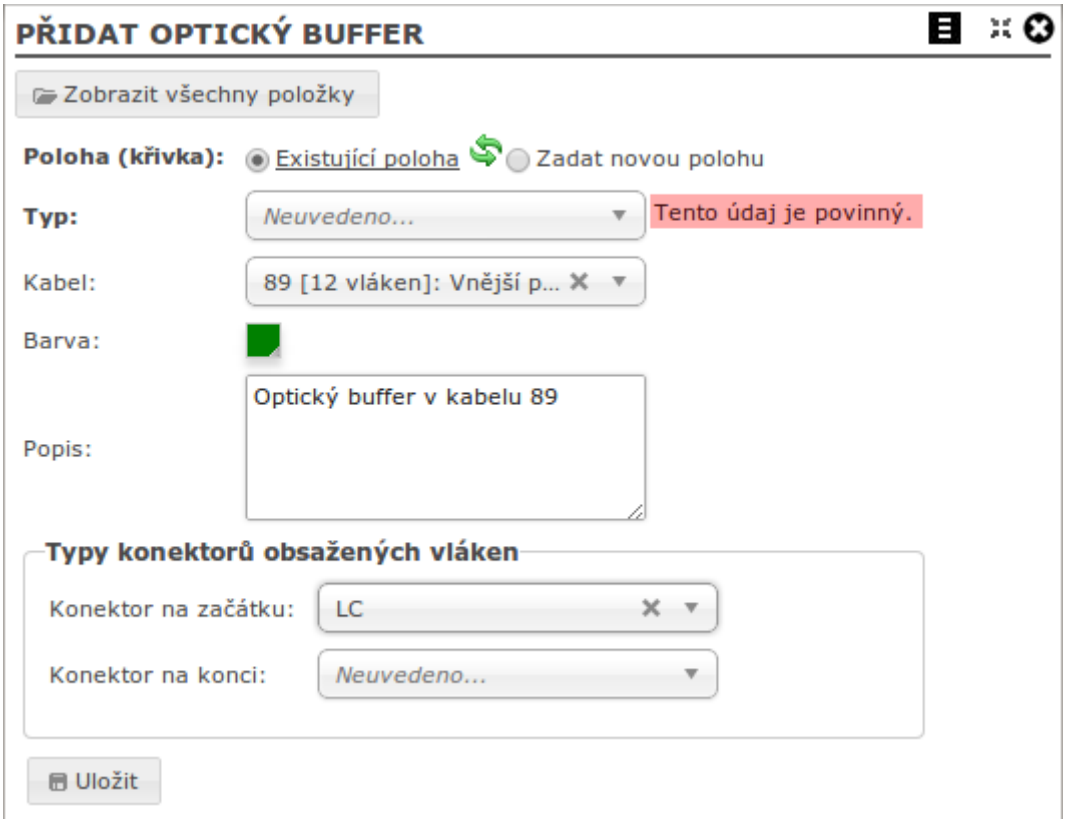


Obrázek C.8: Vyžádání výběru existující pozice na mapě pro vytvoření kabelu na této pozici

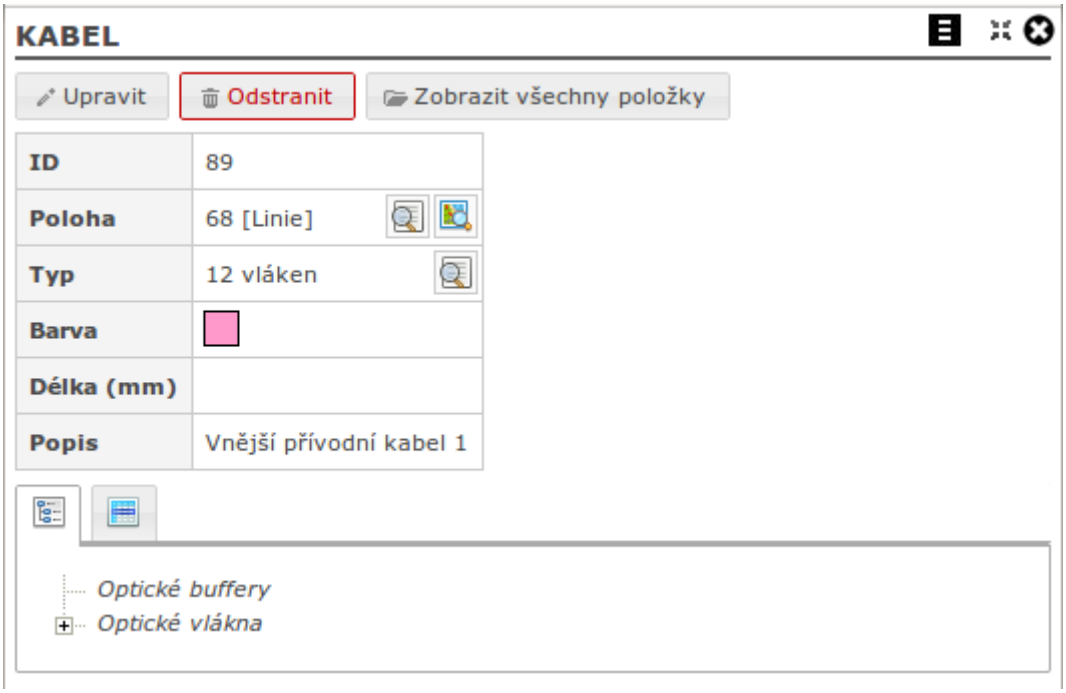
C.1 Snímky klientské aplikace



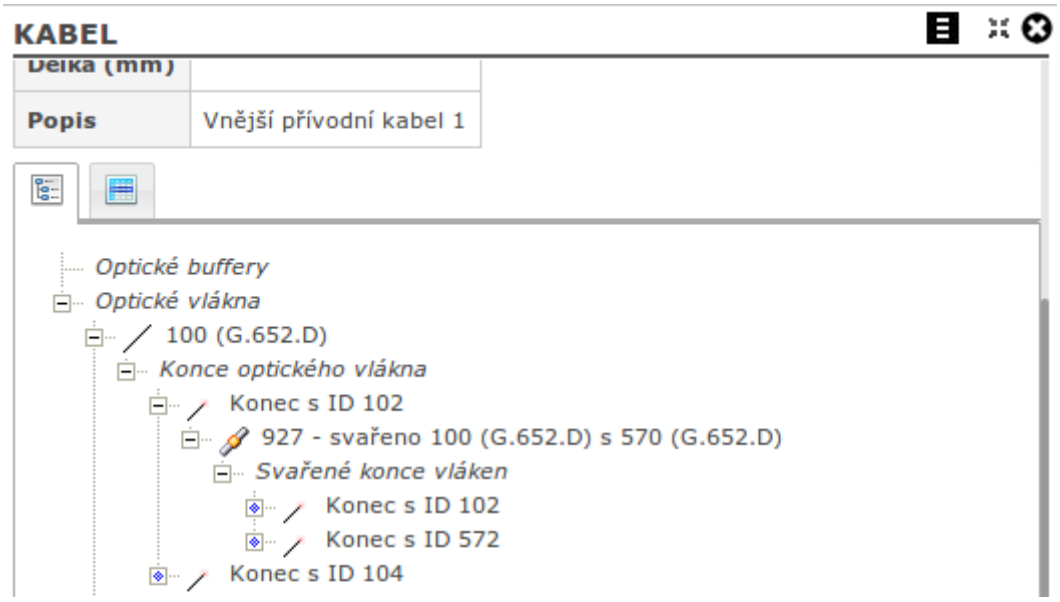
Obrázek C.9: Snímek obrazovky s tabulkou pro zobrazení všech kabelů vytvořené pomocí komponenty dataGrid



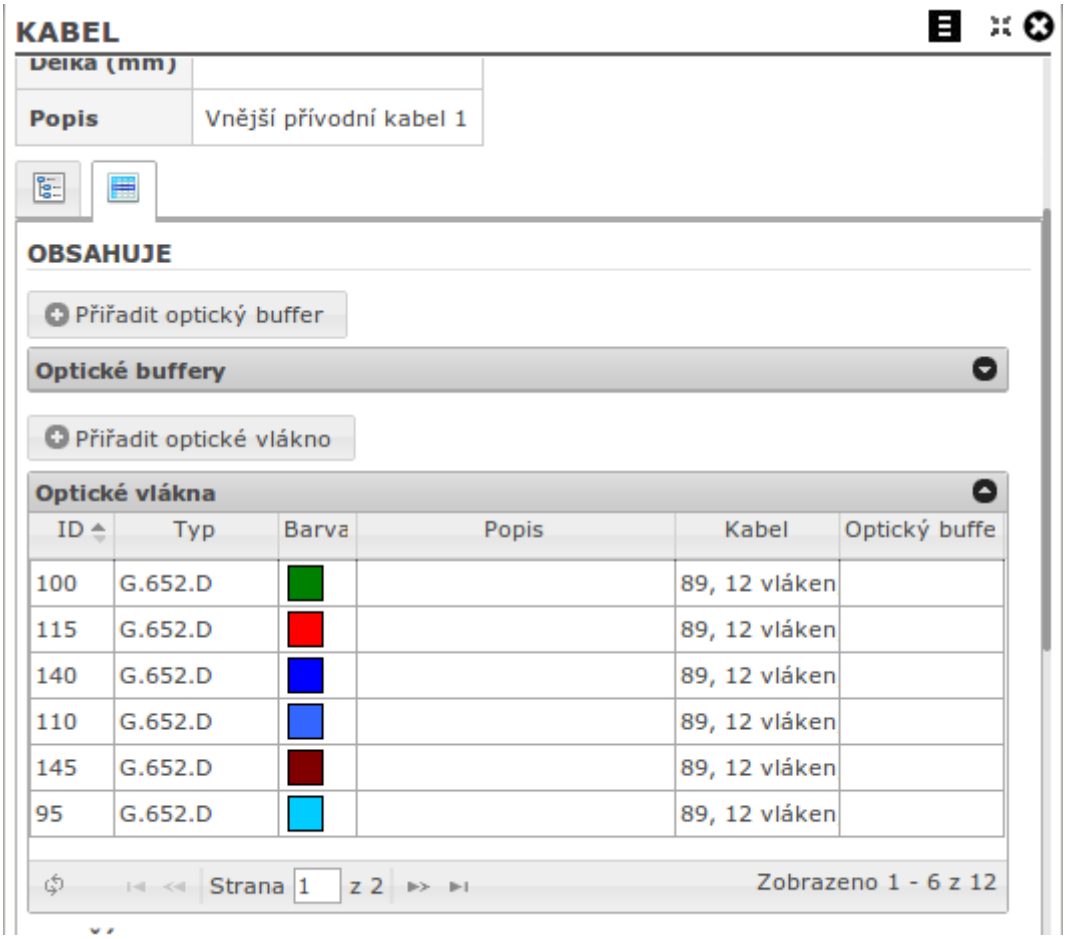
Obrázek C.10: Snímek obrazovky s formulářem pro úpravu kabelu vytvořeným pomocí komponenty dataAdd



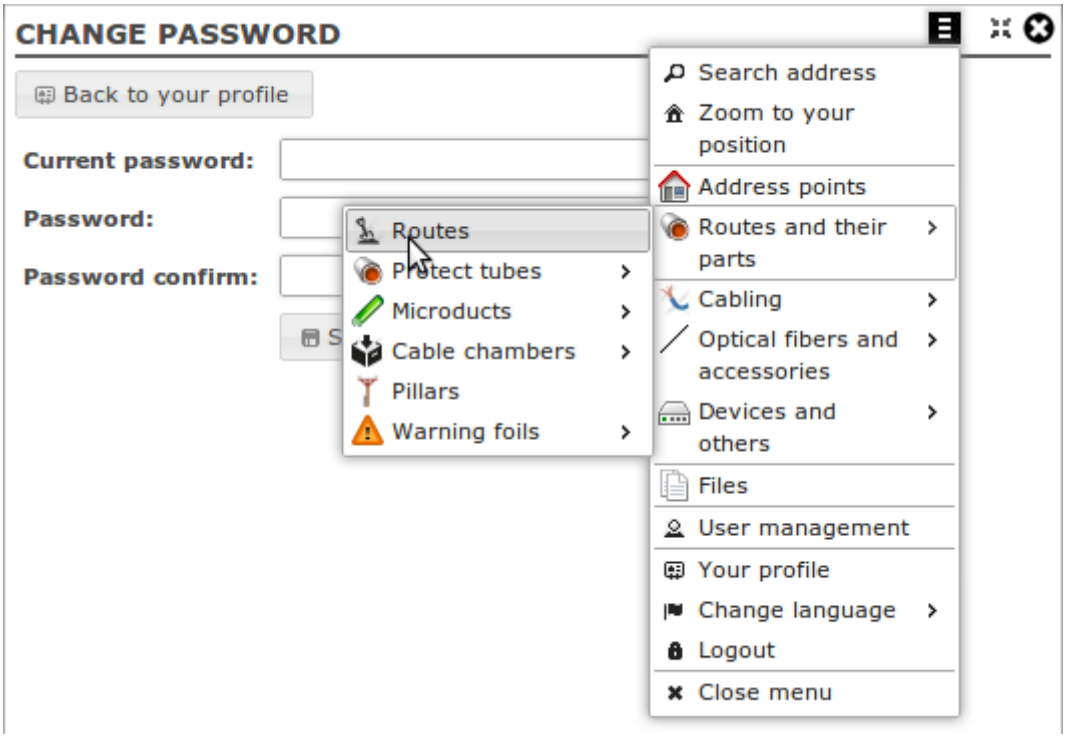
Obrázek C.11: Snímek obrazovky s detailem kabelu vytvořeným pomocí komponenty dataView



Obrázek C.12: Snímek obrazovky se stromovou strukturou síťových prvků návazných na zobrazený kabel



Obrázek C.13: Snímek obrazovky s tabulkami návazných síťových prvků na zobrazený kabel



Obrázek C.14: Internacionalizované uživatelské rozhraní – anglický jazyk